VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Robot založený na Embedded Fusion Tahoe Kit

Robot based on Embedded Fusion Tahoe Kit

2013 **Jakub Juráň**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student:	**Bc. Jakub Juráň**
Studijní program:	N2647 Informační a komunikační technologie
Studijní obor:	2612T025 Informatika a výpočetní technika
Téma:	Robot založený na EmbeddedFusion Tahoe Kit Robot based on EmbeddedFusion Tahoe Kit

Zásady pro vypracování:

V rámci diplomové práce diplomant sestaví robota, který bude využívat EmbeddedFusion Tahoe Kit. Tento kit je programovatelný pomocí .NET Micro Frameworku za pomocí programovacího jazyka C#.

Diplomová práce bude obsahovat níže uvedené body. Diplomant popíše aktuální stav technologie .NET Micro Framework. Dále provede experimentální ověření funkčnosti vytvořeného robota a experimenty zhodnotí.
Jednotlivé body diplomové práce:
1. Konstrukce robota (kostra, motory, převodovka, kola - rozvržení a upevnění prvků).
2. Instalace Tahoe Kit a spojení s pohonným systémem.
3. Vytvoření základních programových ovládacích prvků pro pohyb (.NET Micro Framework).
4. Konstrukce základního rozeznávacího rozhraní pro pohyb v prostoru (IR čidla).
5. Vytvoření programových ovládacích prvků pro pohyb robota pomoci IR čidel.
6. Seznámení se s pohybem v prostoru a programovaného ovládání modulu robota.
7. Konstrukce držáku kamery nebo kamer pro detekci objektů pomocí obrazu nebo při použití dvou kamer stereovize.
8. Implementace programových ovládacích prvků pro zpracování obrazu a následná orientace pohybu v prostoru podle detekovaných objektů.

Seznam doporučené odborné literatury:

[1] .NET Micro Framework: http://www.microsoft.com/netmf/default.mspx
[2] Václav Svatoň , Microsoft Robotic Studio a .NET Micro Framework, diplomová práce, 2010
[3] http://informatix.miloush.net/microframework/Home.aspx

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Martinovič, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7.5.2013 Podpis Jakub Jurák

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu diplomové práce Ing. Janu Martinovičovi Ph.D., za jeho odbornou pomoc, ochotu a trpělivost s jakou mě při tvorbě této diplomové práce vedl. Dále bych chtěl poděkovat panu Lumíru Nedělkovi za pomoc při strojním zpracování konstrukčních prvků a cenné rady. Na závěr chci poděkovat Mgr. Michaele Navrátilové za kontrolu pravopisu a stylistické úpravy práce.

Abstrakt:

Diplomová práce se zabývá konstrukcí robota, který se bude umět orientovat v prostoru na základě detekce objektů, při použití EmbeddedFusion Tahoe Developers Kit jako řídící jednotky. Seznamuje s technologickými prvky .NET Micro Framework a Embedded systémy, které jsou nutné pro vývoj aplikací na této platformě. Následně se konkrétně zaměřuje na konstrukci mechanického systému robota a jeho pohybové prvky. Dále popisuje vývoj programových struktur nutných k ovládání mechanických prvků robota. Na závěr popisuje konstrukci čidel pro detekci objektů a vývoj algoritmů potřebných k analýze informací získaných čidly. Výsledkem práce je popis procesu vývoje robota a jeho samotná funkční konstrukce.

Abstrakt:

This thesis deals with the construction of robot which will be able to navigate in space by detecting objects using EmbeddedFusion Tahoe Developers Kit as a control unit. It introduces the technological elements .NET Micro Framework and Embedded systems which are necessary for the development of applications on this platform. Subsequently it specifically focuses on the construction of a mechanical robot and its motion features. It also describes the evolution of program structures necessary to control the robot's mechanical elements. At the end it describes the construction of sensors for object detection and development of algorithms necessary to analyze the rinformation gathered by sensors. The result of this work is to describe the development process of the robot and its the very functional construction.

Klíčová slova:

EmbeddedFusion Tahoe Developers Kit, .NET Micro Framework, Embedded systémy, robot, čidla, detekce, objekt

Key Words:

Keywords: EmbeddedFusion Tahoe Developers Kit. NET Micro Framework, Embedded systems, robot, sensors, detection, object

Seznam použitých symbolů a zkratek

AA	typ označující velikost a kapacitu baterie – tužkové baterie, kapacita podle složení článku
AAA	typ označující velikost a kapacitu baterie – mikrotužkové baterie, kapacita podle složení článku
CODEC	(CODEC) Kodek je zařízení nebo počítačový program, schopný kódovat nebo dekódovat digitální datový tok nebo signál.
CIF	Common Intermediate Format - standartbí označení zobrazení v rozlišení 128 x 96 (QCIF 176 x 144)
CPU	Procesor též CPU (Central Processing Unit) je základní součástí počítače. Jde o velmi složitý sekvenční obvod, který vykonává strojový kód uložený v operační paměti počítače.
DRAM	DRAM (Dynamic Random Access Memory) je druh počítačové paměti, která uchovává data v podobě elektrického náboje v kondenzátoru
DSP	Digitální signálový procesor nebo také digitální signální procesor (zkratka DSP) je mikroprocesor, jehož návrh je optimalizován pro algoritmy používané při zpracování digitálně reprezentovaných signálů. Hlavním nárokem na systém bývá průběžné zpracování velkého množství dat „protékajících“ procesorem.
GPIO	General Purpose Input/Output (GPIO) je obecný pin čipu mikrokontroléru, jehož chování (včetně toho, zda se jedná o vstupní nebo výstupní pin) je ovládáno programem době jeho běhu.
I2C	Sběrnice I2C (I2C-bus, Inter-IC-bus, Inter-Integrated Circuit) je dvouvodičové datové propojení mezi jedním nebo několika procesory (Masters) a speciálními periferními součástkami (Slaves).
IGBT	bipolární tranzistor s izolovaným hradlem (Anglicky Insulated Gate Bipolar Transistor IGBT) je druh tranzistorů, který je zkonstruován pro velký rozsah spínaných výkonů (od zlomků W až po desítky MW) a vysokou pulzní frekvenci.
IO	integrovaný obvod
IR	Infračervené záření (také IR, z anglického infrared) je elektromagnetické záření s vlnovou délkou větší než viditelné světlo, ale menší než mikrovlnné záření.
JPEG	standardní metoda ztrátové komprese používané pro ukládání počítačových Obrazů

LED	LED (z anglického Light-Emitting Diode - dioda emitující světlo) je polovodičový světelný zdroj.
LIDAR (LADAR)	LIDAR (Light Detection And Ranging, také LADAR) je metoda dálkového průzkumu měření vzdálenosti na základě výpočtu rychlosti odraženého pulsu laserového paprsku od snímaného objektu.
MOSFET	Metal Oxide Semiconductor Field Effect Transistor je polem řízený tranzistor , kde je vodivost kanálu mezi elektrodami Source a Drain ovládána elektrickým polem vytvářeným ve struktuře kov(M)- oxid(O)-polovodič(S) napětím přiloženým mezi hradlo (Gate) a Source. Hradlo je odděleno od polovodiče vrstvou oxidu křemíku – odtud oxid v názvu tohoto typu tranzistoru.
NPN	Uspořádání použitých polovodičů typu P nebo N se rozlišují dva typy bipolárních tranzistorů, NPN a PNP
PWM	pulsně šířková modulace, neboli PWM (Pulse Width Modulation) je diskrétní modulace pro přenos analogového signálu pomocí dvouhodnotového signálu.
RAW	RAW (z anglického raw, což je surový, nezpracovaný) je soubor obsahující minimálně zpracovaná data ze snímače digitálního fotoaparátu. RAW není přímo souborový formát, ale spíše třída (či klasifikace) souborových formátů, protože každý výrobce implementuje jiný formát RAW souborů.
RGB	Barevný model RGB neboli červená-zelená-modrá je aditivní způsob míchání barev používaný ve všech monitorech a projektorech
RISC	RISC (anglicky Reduced Instruction Set Computing, výslovnost risk) označuje v informatice jednu z architektur mikroprocesorů. RISC označuje procesory s redukovanou instrukční sadou.
RS232(COM)	Standard RS-232, resp. jeho poslední varianta RS-232C z roku 1969, (také sériový port nebo sériová linka) se používá jako komunikační rozhraníosobních počítačů a další elektroniky.
SCI	(Serial Communication Interface) Sériové asynchronní rozhraní typu point-topoint.
SDRAM	SDRAM je zkratka z anglického Synchronous Dynamic Random Access Memory, označující paměť typu DRAM se synchronním způsobem přenosu dat.
SDK	Software Development Kit, což je kompletní sada nástrojů pro vývoj software pomocí různých programovacích jazyků, jako například Visual Basic, Visual C++, C#. SDK zahrnuje DLL knihovny, a detailní popis těchto knihoven.

SIF	SIF (Source Input Format) formát zobrazení převzatý z MPEG-1 při rozlišení 352x240
SMD	Surface Mount Device - součástka pro povrchovou montáž plošných spojů
SPI	Seriál Peripheral Interface, sériová sběrnice pro komunikaci mezi mikroprocesory a integrovanými obvody.
TFT LCD	(Thin-film-transistor liquid-crystal display) je varianta displaye z tekutých krystalů (LCD), která využívá tenký film tvořený tranzistory pro ovládání jednotlivých obrazových bodů (TFT). Tato technologie umožňuje zlepšení kvality obrazu, snadnější adresování jednotlivých bodů a dosažení lepšího kontrastu.
TOF	Time of flight (TOF) - jedná se o metodu, která určuje vzdálenost objektu podle měření rychlosti letu elektromagnetického nebo světelného vlnění..
UART	Universal Asynchronous Receiver/Transmitter, hardware který převádí data z paralelních na sériová nebo ze sériových na paralelní.
USB	(Universal Serial Bus) je univerzální sériová sběrnice, moderní způsob připojení periferií k počítači.
VGA	Video Graphics Array (VGA) je počítačový standard pro počítačovou zobrazovací techniku, vydaný roku 1987 společností IBM.
WiFi	Wi-Fi (nebo také Wi-fi, WiFi, Wifi, wi-fi, wifi) je v informatice označení pro několik standardů IEEE 802.11 popisujících bezdrátovou komunikaci v počítačových sítích(též Wireless LAN, WLAN). Samotný název WiFi vytvořilo Wireless Ethernet Compatibility Aliance.
YUV	je barevný model používaný v televizním vysílání v normě PAL i HDTV. Model k popisu barvy používá tříprvkový vektor [Y,U,V]

Obsah

1. ÚVOD

Výsledkem této práce má být funkční sestavení robota, který bude reagovat na okolní prostředí a tomuto prostředí uzpůsobovat svůj pohyb. K implementaci robota byl vybrán modul EmbeddedFusion Tahoe Developers Kit jako řídící modul celého robota. Tento modul je provozován na běhové platformě .Net Micro Framework od společnosti Microsoft.

Běhové prostředí Microsoft .Net Micro Framework [1] kombinuje spolehlivost a účinnost spravovaného kódu a špičkových prostředků vývojových nástrojů Microsoft Visual Studio s možností vyvíjet aplikace pro sofistikovaný hardware malých řídících systémů s jinou architekturou než je architektura PC. Tyto vývojové nástroje zajišťují vynikající produktivitu při vývoji embedded aplikací pro malá zařízení. Microsoft. NET Micro Framework SDK [2] podporuje vývoj aplikací pro tato malá tedy ve většině případů mobilní zařízení jako jsou nejrůznější měřící a řídící systémy. Pro tyto systémy je možné vyvíjet aplikace v jazyce C # pomocí velkého spektra .NET knihoven, které jsou součástí vývojových prostředí typu Microsoft Visual Studio. .NET Micro Framework podporuje všechny hlavní namespace a typy proměnných, která se využívají pro vývoj aplikací na standardní platformě PC. Zmiňované knihovny ®. NET Micro Framework podporují také vzdálené aktualizace firmwaru a různé druhy šifrování pro zabezpečení různých způsobů komunikace zařízení.

Embedded systémy[11] jsou jedny z nejrozšířenějších variant v dnešní době využívaných počítačových systémů. Tyto systémy bývají hlavní řídící součástí větších rozsáhlejších systémů. Příklady použití embedded systémů můžeme najít i v běžném životě. Kdo z nás dnes nezná přístroje, jako jsou MP3 přehrávače, tablety, digitální fotoaparáty apod.

Tyto embedded systémy[12] jsou ve většině případů "autonomní", tzn. embedded systémy jsou schopny dlouhodobě vykonávat své funkce a to bez jakéhokoliv vnějšího zásahu člověka. Tím není samozřejmě není myšleno že by jsme u fotoaparátu nemuseli mačkat spoušť, ale spíše jde o to že nám fotoaparát automaticky ostří, že má funkce odstranění červených očí, automatickou detekci obličeje apod. . Samozřejmě se neobejdeme bez výměny nebo nabíjení baterií, ale to je prováděno jednou za delší časový interval což je v souladu s definicí týkající se embedded systémů.

Další klíčovím požadavkem většiny embedded systémů je, že by měly být neviditelné (skryté), tj. uživatel by je neměl považovat za počítač.

U výše zmíněného fotoaparátu je to jeho řídící jednotka, která se stará o všechny výše zmíněné funkce (ostření, automatická detekce obličeje apod.). Ta je skryta před zraky uživatele v plastovém krytu fotoaparátu.

Jak jste si určitě všimli všechna výše jmenovaná zařízení jsou mobilní a fungují bez přímého připojení k energetické síti, proto je při vývoji embedded systémů kladen obzvláštní důraz na jejich energetickou spotřebu.

Embbeded systémy, ale nemusí být jen systémy mobilními a přenosnými. Mohou být také součástí velkých nemobilních zařízení , jako jsou například halové výrobní linky, kde ve většině případů tvoří hlavní řídící jednotky. I tady je samozřejmě dbáno na co nejnižší energetickou náročnost systému.

Nedílnou součástí požadavků na tyto systémy je také jejich robustnost a spolehlivost.

Autonomní činnost je nezbytná především tam, kde reakce člověka mohou být příliš pomalé, nedostatečně předvídatelné nebo nežádoucí. Systémy, které pracují v reálném čase jako jsou výrobní linky, musí být velmi rychlé při vykonávání daných funkcí a přesné, aby byl například při sváření přivařen díl na místo, kam patří.

Embedded systémy jsou tedy takové elektrotechnické celky, do nichž je neoddělitelně vestavěn mikrokontrolér a jejichž funkce je jeho správnou činností podmíněna. Jedná se tak o neoddělitelné spojení hardwaru a softwaru.

V dnešní době je přibližně 90% všech procesorů využíváno právě jako součást embedded systémů. I když je možné embedded systémy rozdělit na hardwarovou a softwarovou část, není možné použít klasický model návrhu a testování, kdy se tyto dvě větve testují odděleně. V případě vestavěných systémů se objevují nové problémy a vzájemné propojení hardware a činnosti programu. Klasickým příkladem vestavěného systému je mobilní telefon, v němž integrovaný kontrolér řídí i samotné zajištění správných napájecích napětí.

Embedded systém si tedy jednoduše představme jako malý plošný spoj se součástkami, umístěný někde v jádru celého systému. Tento malý plošný spoj se stará o chod tohoto systému. Dá se říci, že je mozkem celého systému, nebo jeho součástí. Systémem v tomto případě myslíme jak velkou výrobní linku, tak i například výše zmíněný digitální fotoaparát. Dále si představme, že Embedded systém, malý plošný spoj někde uprostřed systému, je spojen vodiči se strukturami, které v systému řídí.

Tímto dosáhneme ve výsledku rámcové představy co Embedded systém je a kde jej můžeme ve větším systému najít.

V této práci se budu snažit pomocí takového Embedded systému vytvořit robota.

Jeho úkolem bude pohybovat se v prostoru a bude se snažit vyhýbat předmětům, které potká a které budou leže v trase jeho jízdy. Úkolem bylo tedy tímto Embedded systém řídit pohybový systém robota, získávat informace z detekčních zařízení o stavu okolí a tyto informace vyhodnocovat.

2. EMBEDDEDFUSION TAHOE DEVELOPERS KIT

V této kapitole budou popsány parametry EmbeddedFusion Tahoe Developers Kit, jeho periferie a jeho vnitřní struktura. Dále zde budou uvedeny základní jak programové tak i hardwarové aplikace, které přispěli k ověření funkcí EmbeddedFusion Tahoe Developers Kit.

2.1. EmbeddedFusion Tahoe Developers Kit

EmbeddedFusion Tahoe Developers Kit [3], dále jen Tahoe Kit, je vývojový kit určený pro tvorbu aplikací pomocí technologií firmy Microsoft a to především platformy .Net Micro Framework. V kombinaci s podporou vývojového prostředí, které poskytuje Visual Studio firmy Microsoft a nejmodernějších programovacích jazyků (C#, VB, C++ …) je vývoj aplikací pro podporované mikroprocesory a Embedded systémy dostupné každému programátorovi. Pokud se již programátor někdy alespoň malou částí podílel na vývoji aplikací pomocí vývojových nástrojů firmy Microsoft, bude pro něj vývoj aplikací pro tyto Embedded systémy o to snadnější.

Tahoe kit (Obr.29, Obr.30 v příloze na CD obsahuje mnoho vstupních a výstupních portů, sběrnic a pinů, určených ke snadnému připojení a následné komunikaci s celou škálou externích zařízení, jako jsou například relé, zobrazovací moduly, teplotní čidla apod.

Srdce Tahoe kitu tvoří Meridian CPU modul. Tento modul obsahuje procesory firmy Freescale Semiconductors i.MXS nebo i.MXL ARM920T ™. Modul je konstruován tak, aby byl snadno implementovatelný do koncového zařízení a vzhledem ke své velikosti, nezabral příliš místa.

Meridian CPU modul obsahuje [14] :

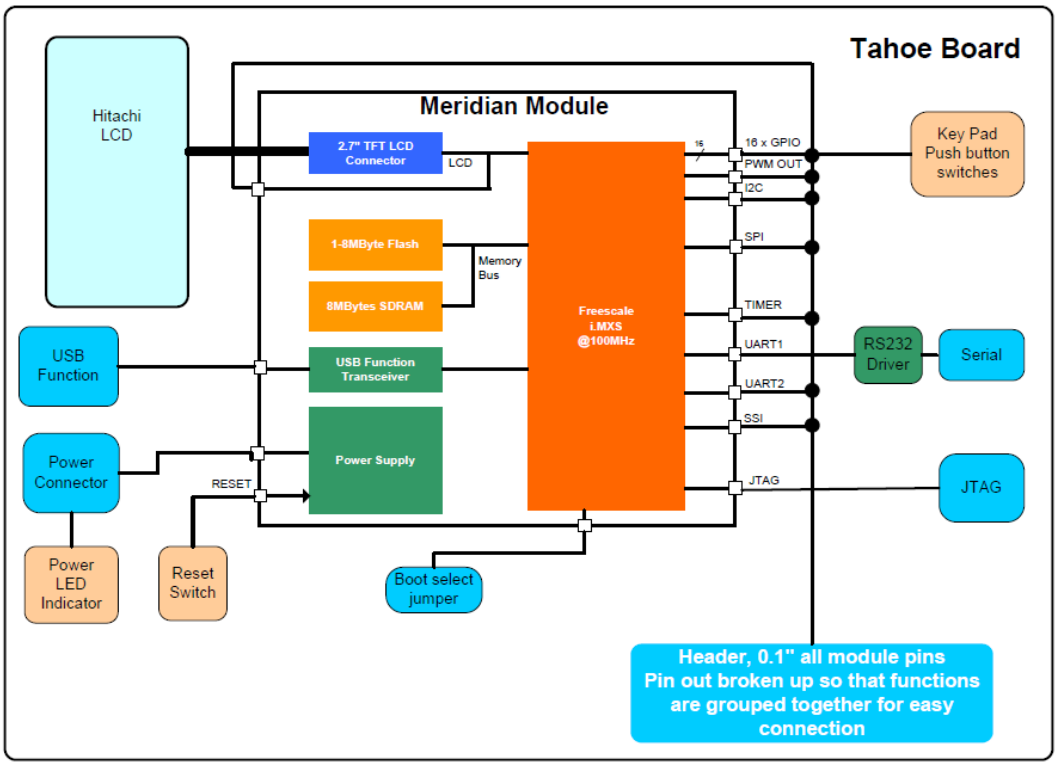
- Freescale processor 100-Mhz typu i.MXS nebo i.MXL (ARM920T)
- 8 MB paměti SDRAM
- 1-8 MB flash paměti
- Konektor pro 2,7 palcový QVGA (320x240) TFT LCD panel
- 16 až 32 GPIO programovatelných portů
- SPI sběrnici
- I2C sběrnici
- PWM kanál

- USB port pro vývoj a zavádění aplikací
- Vstup napájecího napětí (5V) - modul může být napájen z výše zmíněného USB portu
- Výstup 3.3V pro připojení zařízení s jinou napěťovou hladinou, jak napájecí tak logickou

Všechny tyto I/O porty jsou na malém Meridian CPU modulu situovány do jednoho microkonektoru, pomocí kterého se potom Meridian CPU modul připojuje k ovládanému zařízení. Není proto možné reálně vyvíjet a testovat aplikace a připojovat je přes zmíněný mikrokonektor.

K účelům vývoje aplikací pro Meridian CPU modul proto slouží Tahoe kit, který má všechny programovatelné porty a piny přístupné a není proto žádný problém připojit jakékoliv zařízení.

Díky této modulárnosti je možné pomocí Tahoe Kitu vyvinout aplikaci, kterou lze řádně odzkoušet na přehledné desce Tahoe Kitu a posléze využít jen jádro, tedy Meridian CPU modul. Na tomto naprogramovaném modulu již využíváme jen I/O portů potřebných k fungování vytvořené aplikace. Tímto se nám sníží požadavky na prostor, vzhledem k velikosti modulu a případné redukci počtu vývodů z Meridian CPU modulu jen na vývody využívané aplikací. Následující obrázek reprezentuje propojení Meridian CPU modulu a Tahoe Kitu .



Obr. 1. Názorná ukázka propojení Meridian CPU modulu a samotného Tahoe Kitu .

Seznam a popis jednotlivých periferií, které jsou graficky znázorněny na obrázku (Obr.1) a pomocí kterých lze s Tahoe kitem komunikovat [14].

LCD - LCD display je integrován přímo na desce Tahoe Kitu, ale také je ožné připojit v případě nutnosti i display externí. Vývody k tomuto účelu jsou na desce k dispozici.

USB - Funkce USB portu Meridian CPU modulu jsou na desce reprezentovány pomocí USB mini-B konektoru. USB port v tomto případě slouží jako již zmíněné alternativní napájení celého Tahoe Kitu a zároveň pro připonejí k PC pro účely zavádění a testování aplikací přímo z Visual Studia.

Keypad - Klávesnice tvořená devíti mikrospínači. Mikrospínače jsou napojeny přímo na Switches piny GPIO. Návrh osazení jednotlivých mikrospínačů/tlačítek vychází z nejběžnějšího uložení tlačítek na různých mobilních zařízeních.

RS232 - UART1 výstup z Meridian CPU modulu je připraven pro komunikaci pomocí protokolu RS232 na desce Tahoe Kitu konektoru DB9

Power Indicator - Tahoe Kit disponuje jednou zelenou LED diodou pro signalizaci přivedeného napájecího napětí.

Boot Mode - Tahoe Kit má možnost zavádět aplikace ze dvou zdrojů. Těmito zdroji jsou Header Flash paměť, do které se zapisuje zaváděný program z Visual Studia a ze sériového portu pomocí kterého můžeme zvyšovat verze zaváděcího programu v Meridian CPU modulu. K přepínání nám slouží jednoduchý přepínač umístěný na desce Tahoe Kitu.

Reset Switch - Toto tlačítko resetuje Tahoe Kit - provede vymazání paměti programu bez nutnosti odpojení napájecího napětí. Při lazení programu pomocí prostředí Visual Studio není nutné ručně Tahoe Kit restartovat. Restart a následné nahrávání programu je prováděno automaticky.

Power Supply - Konektor napájecího napětí 5V/2A .

Expansion - Jednotlivé programovatelné piny GPIOConnectors

Pin Interface - Mikrokonektor s 80ti piny zmiňovaný již výše, který propojuje Meridian CPU modul a Tahoe Kitem.

2.2. Microsoft Visual Studio a EmbeddedFusion Tahoe Kit

Pro vytváření aplikací pomocí .NET Micro Frameworku a Tahoe kitu jen´nutné mít nainstalovány následující komponenty (uvedená konfigurace byla použita pro vývoj robota) :

- Visual Studio 2008
- .NET Framework verze 3.5
- .NET Micro Framework verze 3.0
- DeviceSolutionsSDK (SDK pro vývoj aplikací na Tahoe kitu)

Po spuštění nástroje Visual Studio se nám od této chvíle v nabídce pro vytvoření nového projektu nabízí volba projektu Micro Framework. V nabídce Templates si potom navolíme o jaký typ projektu půjde.

Možnosti jsou následující :

- Console Application
- Window Aplication
- Class Library
- Device Emulator.

Pro vývoj testovacích aplikací robota bylo nejvíce využíváno projektů Console Application a Class Library.

Dále můžeme aplikace vyvíjet přímo pro zařízení jako je Tahoe kit, které je připojené k počítači pomocí USB portu, nebo můžeme vyvíjet aplikace s využitím emulátoru. Volba zařízení je dostupná v Properties otevřených nad projektem .NET Micro Frameworku v záložce .NET Micro Framework. Zde jsou v oblasti Deployment dostupné volby Transport a Device. Ve volbě Transport si můžeme vybrat možnosti připojení zařízení, pro které budeme vyvíjet (USB, TCP/IP, RS232). Je-li zařízení detekováno, je okamžitě nabídnuto ve volbě Device.

Jinou variantou výběru je možné použít v nabídce Transport volbu Emulator a v tom případě nám volba Device nabízí tři emulátory TahoeEmulator, Tahoe-II Emulator a Microsoft Emulator.

V našem případě byla zvolena ve volbě Transport možnost komunikace pomocí USB a následně pokud byl Tahoe kit připojen k počítači bylo hned v nabídce Device nabídnuto zařízení Meridian_a7e70ea2. Po uložení této volby se při každém zpuštění debugingu(F5) ve Visual Studiu provedlo vložení debugovaného kódu přímo do Tahoe kitu a jeho spuštění. V praxi se jedná o největší výhodu jakou nám může takovýto vývoj poskytnout tzn. ověřit si funkcionalitu kódu přímo na fyzickém zařízení, pro které je aplikace vyvíjena.

2.3. Základní programové konstrukce

Při seznamování se s Tahoe kitem a jeho možnostmi byly sestaveny základní komponenty, které poskytly možnost otestovat různá zapojení a naprogramování Tahoe kitu. Seznam použitých komponent, které byly pro účely testování zkonstruovány : Servomotor, reproduktory, LED dioda zelená, LED dioda červená, Modul LED diod (RYG), tlačítka - mikrospínače, display, relé, JPEG kamera, IR senzory, stejnosměrné motory.

Pro vytváření programových ovládacích prvků byly použity jmenné prostory, které jsou součástí .Net Micro frameworku (Microsoft.SPOT) a další jmenné prostory, které jsou součástí SDK Tahoe kitu (DeviceSolutions.SPOT).

Ve jmenném prostoru , DeviceSolutions.SPOT.Hardware SDK Tahoe Kitu se nalézá třída OutpuPort, která reprezentuje práci se všemi jednotlivými programovatelnými piny GPIO. Tento jmenný prostor a tato třída bude jednou ze stěžejních částí celé programové konstrukce robota.

Další třídy a jmenné prostory, nutné k běhu programu na mikroprocesoru jsou Microsoft.SPOT.Native, Microsoft.SPOT.TinyCore, Microsoft.SPOT.Hardware. Popisy funkcí jednotlivých knihoven [15] lze nalézt na stránkách firmy Microsoft, proto se dále budu zabývat pouze třídami, které byly při vývoji používány.

Programová práce s jednotlivými piny není nikterak složitá, proto budou v následující části uvedeny jen dva stručné příklady.

Jeden zaměřený na výstup - rozsvícení a zhasnutí led diody v intervalu 1000 ms.

A druhý zaměřený na vstup, kontrola stisknutí tlačítka. Podobným způsobem se budou dále ovládat motory a očekávat změna stavu IR čidla, ale to je popsáno až v následujících částech práce.

2.4. Definice pinu a portu

Pro ujasnění dalšího výkladu je nutné upřesnit význam slova "pin" a slova "port". Pokud hovoříme o "pinu" hovoříme o fyzickém vývodu přímo na modulu na Tahoe kitu, který bude používán v souvislosti s elektronickým zapojením. "Porty" rozumíme inicializované "piny" v programu, kterým byla přiřazena programová funkce a to funkce vstupní, nebo funkce výstupní. Funkcí výstupní rozumíme nastavování napěťových úrovní pinu programově a funkcí vstupní rozumíme nastavování napěťové úrovně připojeným zařízením k tomuto pinu.

2.5. Práce s piny a porty

Začneme tedy výstupním řízením pinu. Jako první bod, který je nutno provést je zvolit si pin na kterém budeme provádět změnu stavu. Sběrnice GPIO obsahuje 12 programovatelných pinů, které budeme používat.

Změnou stavu myslíme změnu ze stavu High, tedy na pinu naměříme 5V na úroveň LOW, na pinu naměříme 0V nebo naopak.

Pro nás bude tímto pracovním pinem pin GPIO1 , který je na modulu Tahoe Kitu označen PB19 sběrnice GPIO. Programově si jej při vytváření pojmenujeme Led_L1.

```
static Cpu.Pin Led_L1 = Meridian.Pins.GPIO1; //PB19
```

Dalším krokem je vytvoření portu a inicializace portu pro zápis. V základní hodnotě LOW (0V), kterou reprezentuje hodnota "false" v konstruktoru objektu OutputPort.

```
public static OutputPort Led1_Port = new OutputPort(Led_L1, false);
```

V metodě main() jejíž výpis následuje, potom zavedeme nekonečný cyklus se zápisem změny stavu na portu Led1_Port a uspíme proces po každé změně stavu na 1000 ms.
Tím dosáhneme střídání stavu (High / Low - 5V / 0V) v sekundovém intervalu. Pokud by jsme připojily k pinu GPIO1 led diodu tak by dioda v sekundovém intervalu blikala.

```
public void Main()
{
        while (true)
        {
                Led1_Port.Write(true);
                Thread.Sleep(1000);
                Led1_Port.Write(false);
                Thread.Sleep(1000);
        }
}
```

Tímto příkladem bylo demonstrováno jak inicializovat pin Tahoe kitu jako výstupní a zapsat na výstupní port.

V následujícím příkladu bude ukázáno jak inicializovat pin jako výstupní a popsáno odchytávání změny stavu na portu při stisku tlačítka.

Opět byl zvolen pin GPIO1 na který bylo tlačítko připojeno.

```
static Cpu.Pin Tlacitko_T1 = Meridian.Pins.GPIO1;
```

V v níže uvedeném kódu byla provedena inicializace portu Tlacitko_P1. Jako první parametr inicializace se v konstruktoru předává zvolený pin Tlacitko_T1 / GPIO1. Další parametr zapíná takzvaný filtr zákmitů - glitch filter. K zákmitům může docházet například u některých druhů tlačítek. Při spínání mechanických kontaktů často dochází k krátkodobému odskočení kontaktů a vzniku několika impulzů, způsobující nesprávné vyhodnocení jednotlivých sepnutí. Proto je v určitých případech nutné takovýmto stavům předejít.

Je možné toto řešit použitím speciálních obvodů zamezujících zakmitávání kontaktů, nebo tyto stavy ošetřit programově. Programovým ošetřením se často používá časové prodlevy mezi vícenásobným čtením stavu spínače. Tuto problematiku, ale přenecháme zapnutím výše zmíněného parametru, plně na modulu Thoe Kitu kde je ošetření proti zákmitům realizováno vnitřně.

Dalším parametrem je nastavení PullUp a PullDown rezistorů. Podle definice [5] "Na každém logickém vstupu musí být definovaná log. úroveň.

V případě nezapojeného vstupu, ke kterému je připojena vysoká impedance, nebo výstupu s otevřeným kolektorem, může dojít k neočekávaným stavům."

Tyto stavy se ošetřují připojením vstupu, nebo výstupu přes rezistor na svorky napájecího napětí a to na kladný nebo záporný pól. Výše zmíněná zapojení jsou znázorněna obrázky v příloze na CD Obr.1 a Obr.2. Opět díky vnitřnímu zapojení pouze vyberu správnou hodnotu parametru ResistorMode, která je typu enum a zbytek nechat na Tahoe Kitu.

Posledním parametrem je nastavení na jakou změnu stavu se má vyvolat událost odchycení změny stavu. Opět parametr InterruptMode je typu enum a můžeme si tedy vybrat z několika předdefinovaných přechodů. Nastavíme změnu stavu z Low (0V) na High (5V), tedy jeden vodič spínače připojíme na napájecí napětí 5V a druhý na námi vybraný pin GPIO1.

```
private static InterruptPort Tlacitko_P1 = new InterruptPort(Tlacitko_T1, true, Port.ResistorMode.Disabled, Port.InterruptMode. InterruptEdgeHigh);
```

V metodě Main() zaregistrujeme odběr události při změně stavu.

```
public void Main()
{
        Tlacitko_P1.OnInterrupt += new NativeEventHandler(port_OnInterrupt_ Tlacitko_P1);
}
```

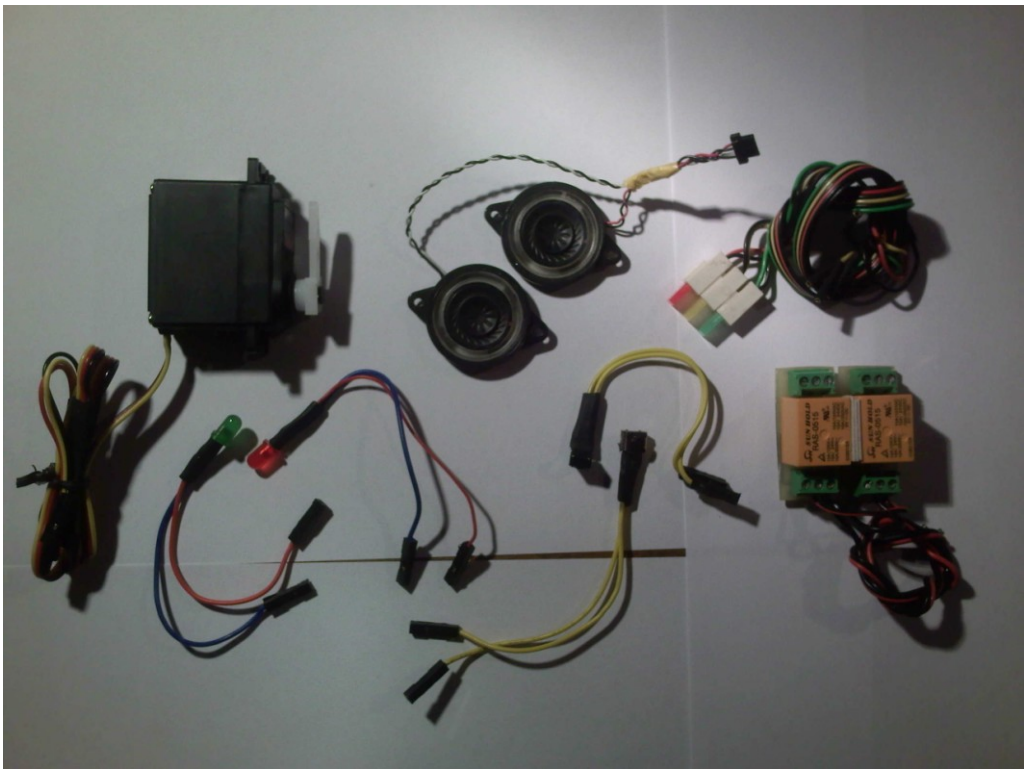
Metoda, která se provede při vyvolání události změny stavu.

```
private void port_OnInterrupt_ Tlacitko_P1 (uint port, uint state, TimeSpan time)
{
        // zapnutí diody, sepnutí relé, apod.
}
```

V případě, že by jsme si inicializovali jiný pin pro led diodu, jako v předchozím příkladě, mohli by jsme, při provádění této metody, která je vyvolána událostí změny stavu, diodu rozsvítit.

Pomocí těchto znalostí týkajících se ovládání pinů Tahoe Kitu jsme schopni realizovat základní úlohy ovládání výše zmíněných komponent.

Jedná se o následující úlohy: blikání LED diody (pomocí uspání procesu nebo časovačem), rozsvícení a zhasnutí LED diody v závislosti na stisku tlačítka, rozsvěcování více LED diod - každá vlastním procesem a časovačem, generování zvuku, otáčení servomotorem - pulsní modulace, spínání žárovky pomocí relé, spínání motorů pomocí relé. Vybrané úlohy jsou uvedeny v příloze v sekci Příloha_src této diplomové práce.



Obr. 2. Testovací komponenty

3. KONSTRUKCE ROBOTA

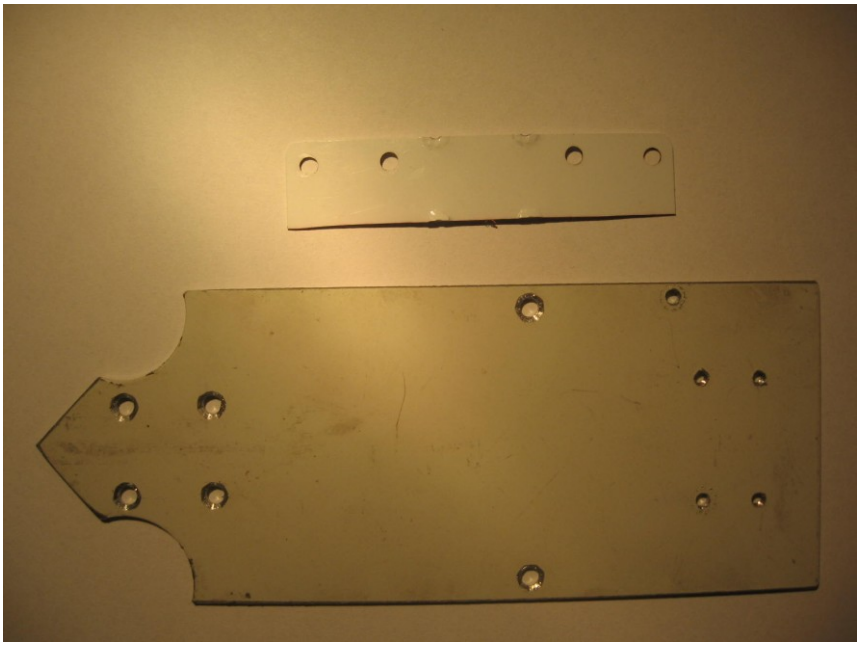
Pro natáčení robota do stran, byl zvolen systém se dvěma velkými na sobě nezávisle zavěšenými koly. Kola jsou umístěnými po stranách robota v jeho středové části a jednou všesměrovou kuličkou na distanční konstrukci v části zadní (dále jen všesměrový pohybový systém).

Natáčení robota je prováděno změnou rychlosti běhu buď levého kola, nebo kola pravého, podle zvoleného směru zatočení. Kvůli všesměrovému pohybovému systému byl sice omezen typ povrchu, po kterém se může robot pohybovat (povrch nemůže být příliš nerovný), ale pro účely vývoje je postačující. Důvod omezení je patrný z konstrukce všesměrového pohybového systému. Hlavní pohybovou částí tohoto systému je ocelová kulička o průměru 0,6 cm. Z toho je patrné že hrubší, nebo nerovnější povrchy by mohl být pro tento typ pohybového systému problémem.

3.1. Kostra

Kostru robota (Obr. 3) tvoří obdélníkový kovový plát o rozměrech 175 x 70 mm. Rozměry plátu jsou upraveny na plochu desky Tahoe Kitu, tak aby přečnívala jen minimálně.

Plát je vyroben z hliníku kvůli jeho snadnému opracování a lehkosti. Dále byly do nosného plátu vrtány díry o průměru 3,1 mm. Dvě v přední části pro uchycení bateriových článků určených k napájení logické části robota. Další dvě rozložené nesouměrně také v přední části nosného plátu pro uchycení Tahoe Kitu. Dvě v části středové pro uchycení dvoumotorové převodovky. Čtyři v zadní části pro uchycení všesměrového pohybového systému. K této poslední části jsou také uchyceny bateriové články pro pohon motorového systému robota. Tyto články mají speciální uchycení na plastový pásek doplněné dodatečně. Původním záměrem bylo robota napájet jen z jednoho bateriového úložiště, což se v průběhu vývoje ukázalo být nedostatečné. Jako spojovací materiál jsou zvoleny šroubky o průměru 3mm a délce 27 mm.



Obr. 3 Celkový pohled na neosazené chasis

3.2. Pohonný systém

Jako pohon robota (Obr. 3 v příloze na CD) byly zvoleny dva stejnosměrné motory, které jsou napájeny 5V. Motory uvádějí do chodu dvojitou čtyřrychlostní převodovku (Obr. 4 v příloze na CD).

Převodové poměry se určují způsobem sestavení převodovky, které nelze za chodu měnit. Možnosti sestavení převodových poměrů jsou: 12.7:1, 38.2:1, 114.7:1, a 344.2:1 Pro převodovku byl zvolen poměr 344.2:1, čímž se docílilo snadnější ovladatelnosti robota při zatáčení.

Kroutící moment převodovky je možné, podle nastavení převodových stupňů nastavit na hodnoty : 9.2 Nm, 27 Nm, 79 Nm, nebo 223 Nm.

Výstupní šestihranná hřídel unášející kola má průměr 3 mm. Rozměry složené převodovky jsou 70x60x23 mm. Montážní otvory mají rozteč 60 mm. Převodovka je umístěna na spodní straně robota mezi držáky baterií.

Použitá kola (Obr.5 v příloze na CD) mají celkový průměr 58 mm, z čehož tvoří 42 mm vnitřní plastové kolečko se středovým otvorem.

Ve středovém prostoru kola je umístěn plastový mechanizmus pro nasazením kola na výše zmíněnou výstupní šestihrannou hřídel převodovky. Na plastovém kolečku je umístěna pneumatika o výšce 16 mm a šířce 16 mm.

4. INSTALACE TAHOE KIT A PROPOJENÍ S POHONNÝM SYSTÉMEM

Vzhledem ke zvolené konstrukci kostry robota nebyla instalace Tahoe Kitu na vlastní podvozek nijak složitá. Všechny komplikace, které při instalaci na samotný podvozek nastaly, byly díky poddajnosti zvoleného materiálu kostry snadno odstraněny.

4.1. Uložení Tahoe Kitu

Tahoe Kit je umístěn z horní strany kostry robota a uchycen dvěma šrouby o průměru 3 mm. Tahoe Kit je ze spodní strany osazen distančními gumičkami, které odbouraly starost o případné krátké spojení při doteku Tahoe kitu s povrchem kostry. Možná se může zdát uchycení pouze dvěmi šrouby nedostatečné, ale díky distančním gumičkám je plošný spoj Tahoe Kitu pevně uchycen.

Tahoe kit byl instalován tak, aby byly všechny potřebné porty (USB port, COM port, napájecí konektor, piny jednotlivých sběrnic) dobře dostupné a umožňovaly snadný přístup při vývoji robota.

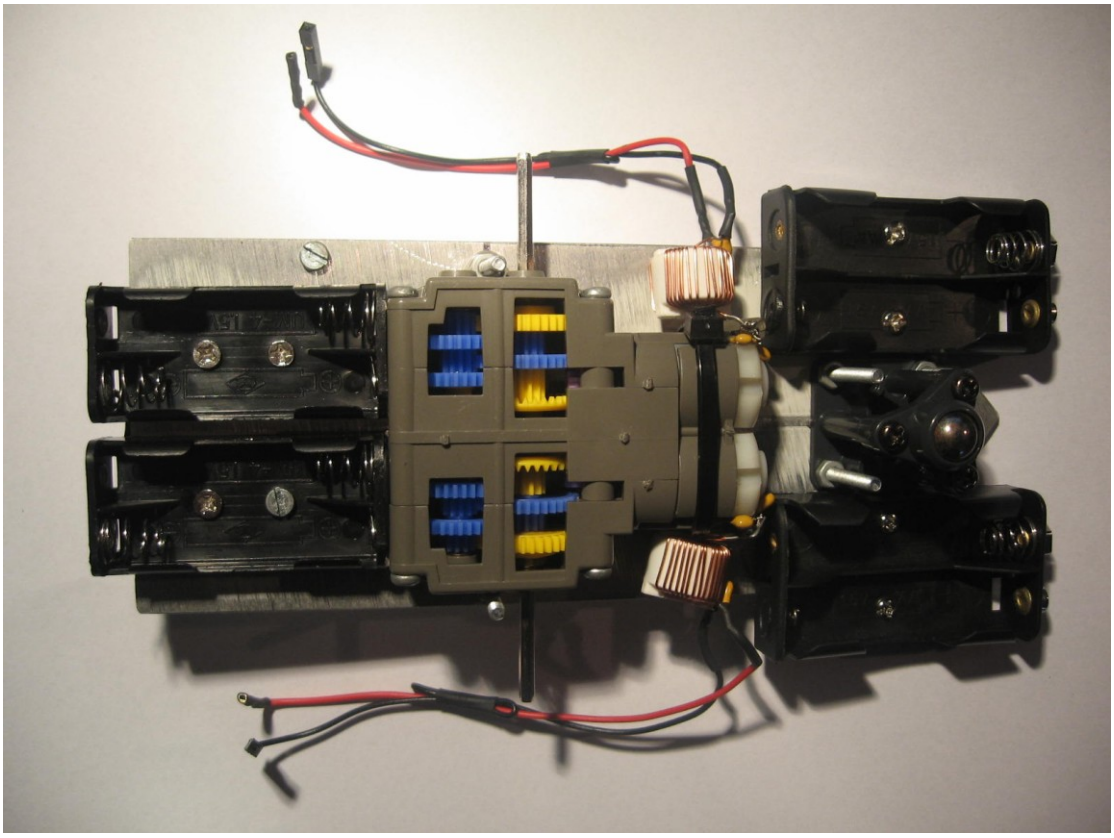
4.2. Uložení pohonného systému a baterií

Robot využívá dvou napájecích bateriových systémů, složených ze čtyř článků baterií typu AAA pro napájení logiky a aplikací s nízkým odběrem proudu a čtyřmi články baterií typu AA pro napájení motorů.

Bateriové držáky (Obr. 6 v příloze na CD) jsou umístěny tak, aby co nejlépe udržovali stabilitu robota a vyvažovali jej, vzhledem k jeho konstrukci (Obr. 7 v příloze na CD). Vzhledem k tomu, že se jedná o konstrukci se třemi styčnými body, z čehož jsou dva hlavní umístěny vprostřed robota a třetí je umístěn na zadní části robota jsou těžší baterie typu AA umístěny v zadní části. Byly uloženy ve spojení po dvou článcích z levé a z pravé strany všesměrového pohybového systému, který je třetím styčným bodem. Lehčí baterie typu AAA jsou uloženy v přední části před převodovkou s hlavími koly, které tvoří první dva styčné body. Tím je zamezeno případnému převážení robota dopředu.

Převodovka(Obr. 8 v příloze na CD) je umístěna, jak bylo zmíněno již výše, v prostřední části robota. Není umístěna přesně v ose souměrnosti celého obdélníku, ale je mírně posunuta do přední části. Rovněž z důvodu zamezení převažování robota.

Převodovka je uložena tak, že delší a těžší část s motory leží směrem k zadní části a část se samotným převodovým soukolím, která je kratší a lehčí leží směrem k přední části.



Obr 4. Uložení pohonného systému a baterií

4.3. Releové ovládání

Základem ovládání pohonné jednotky pomocí relé je jednoduché zapojení tranzistorového spínání relé (Obr. 9, 10 v příloze na CD). Přímé spínání relé Tahoe Kitem není možné z důvodu slabé proudové zatížitelnosti jednotlivých pinů.

Zapojení tranzistorového spínání relé[6] (Obr. 11 v příloze na CD) je určeno pro zařízení, které potřebují malým napětím spínat napětí velká. Výhoda spínání pomocí relé je galvanické oddělení.

Toto zařízení bylo použito hlavně z důvodu toho, že při rozběhu motoru dochází k velkému proudovému odběru a Tahoe kit na jednotlivých pinech a jak bylo zmíněno výše, není schopen toto zatížení zvládnout. Sepnutí kontaktu relé přivede napětí přímo z baterie na vstupní svorky motoru čímž eliminujeme poškození Tahoe kitu, ke kterému by mohlo dojít v případě, že bysme motor připojili přímo na výstupní piny sběrnice Tahoe kitu.

Napájecí napětí máme podle dispozic Tahoe kitu 5V. Hodnoty součástek jsme si vypočítali následujícím způsobem. Zapojení obsahuje pouze jeden tranzistor NPN, použit byl BC547B

ve variantě SMD, ale jde použít jakýkoli podobný typ, jen bude potřeba snížit hodnotu R1, pokud by tranzistor nespínal relé. Pro výpočet R1 potřebujete znát impedanci cívky relé (změříme multimetrem), u použitého zapojení má cívka 300R, proud cívky tedy z Ohmova zákona vychází na 60mA. V dalším kroku je třeba znát proudový zesilovací činitel tranzistoru (HFe,H21). Použitý tranzistor má zesilovací činitel 300. Teď je na řadě vypočítat potřebný proud přiváděný do báze tranzistoru. Vypočítáme tak že proud relé vydělíme proudovým zesilovacím činitelem, vyjde nám 1.3uA , to je malý proud. Poté, už jen stačí vypočítat hodnotu rezistoru R1. Napětí Ube má být 0.7V (saturační napětí), proud 1.3uA, úbytek na rezistoru tedy musí být 4.3V při stanoveném proudu 1.3uA. Hodnotu vypočteme tak že úbytek napětí na rezistoru vydělíme procházejícím proudem, vychází nám přibližně 33K, nejbližší v řadě je odpor 33K a ten byl také použit. Důležitým prvkem v zapojení je také dioda. Cívka relé je indukčnost a i bez diody obvod cívku sepne, vše se zdá být v pořádku, ale není. Když relé vypnete, tak se na cívce se naindukuje vyšší, záporné napětí, které nám tranzistor s největší pravděpodobností nevratně zničí. Dioda zapojená antiparalerně k relé pošle záporné napětí na uzemnění obvodu a naindukované napětí již tranzistor neohrozí. Do zapojení obdobného charakteru se vkládá ještě kondenzátor mezi bázi tranzistoru a zemnění obvodu. Slouží ke krátkému zadržení kotvy relé tím, že udrží na okamžik napětí mezi bází a emitorem. Dokud se tedy nevybije kondenzátor do určité hodnoty, tak zůstane tranzistor a tím pádem i relé sepnuté. Tímto způsobem se zamezuje rychlému spínání a rozepínání relé například při špatně stlačeném tlačítku.

4.4. Změna měna napájení

V původní koncepci (Obr .12 v příloze na CD) bylo plánováno, že celý robot bude napájen jen jedním napájecím systémem se čtyřmi bateriemy typu AAA (jak část logické - Tahoe Kit a čidla, tak část pohonná - motory).

Po prvním testu rozběhnutí motorů pomocí relé, ale došlo k restartu celého Tahoe kitu. To se při každém dalším provádění tohoto testu opakovalo. Celý systém byl podroben měření, ze kterého vyplynulo následující zjištění.

Při sepnutí relé a rozběhu motorů dojde na malý okamžik k velkému poklesu napětí zhruba na 3V což vypne Tahoe kit a tím i celý běh programu. Relé se následně rozpojí a k očekávanému rozběhnutí motorů vůbec nedojde.

Pokles napětí je způsoben velkým proudovým odběrem motorů z baterií při rozběhu. Baterie nejsou schopny takový proud poskytnout, aproto dojde k poklesu napětí. To by bylo možné obejít pomocí kondenzátoru zapojeného na vstupní svorky napájecího napětí Tahoe kitu, který by na krátkou dobu nutnou k rozběhu motorů udržel napájecí napětí pro Tahoe kit na konstantní úrovni. V našem případě byl tento problém vyřešen přidání dalšího napájecího systému jen pro motory. Důvodem byla obava z nedostatečné kapacity stávajících článků. Pro napájení motorů byly zvoleny čtyři větší články typu AA. Díky spínání pomocí relé byly tyto dva napájecí systémy galvanicky odděleny a nehrozilo žádné vzájemné rušení. Při provádění dalších testů běhu motorů bylo již vše v pořádku a motory fungovaly přesně podle instrukcí programu v Tahoe kitu.

Byla ověřena i funkce převodovky a zvoleného pevného převodu, která i přes její hlučnost obstála. Hlučnost způsobují plastová ozubená kolečka převodovky.

Testovací programy měly podobnou strukturu, jaká je zmíněna výše pří seznámení se nastavováním pinů. Jednalo se o seznam po sobě jdoucích instrukcí, které pomocí zápisu logické jedničky nebo logické nuly na pin ovládaly motory.

4.5. Výpis textu na display

Dále bylo otestováno vypisování textu na display Tahoe kitu. Vypisován byl stav chodu motorů tedy jestli jsou v chodu oba motory,nebo jen pravý, nebo jen levý motor. Vypisování textu na display Tahoe Kitu bylo prováděno následujícím způsobem :

Byl vytvořen nový objekt typu Bitmap se šířkou a výškou displaye podle typu zařízení na kterém je prováděn vývoj.

Bitmap MyBitmap = new Bitmap(SystemMetrics.ScreenWidth, SystemMetrics.ScreenHeight);

Voláním následující metody inicializujeme výstupní text pro vypsání na display. Prvním parametrem této metody je text, který chceme vypsat na display, druhým parametrem je zvolený font písma, třetím parametrem je barva písma a na závěr čtvrtým a pátým parametrem jsou souřadnice, na které má být text umístěn (souřadnice bodu, ve kterém bude začínat první písmeno textu).

```
MyBitmap.DrawText("výstupní text",
                  Resources.GetFont(Resources.FontResources.small), Colors.White, 10, 25);
```

Tato instrukce provede vykreslení výsledného obrazu na display.

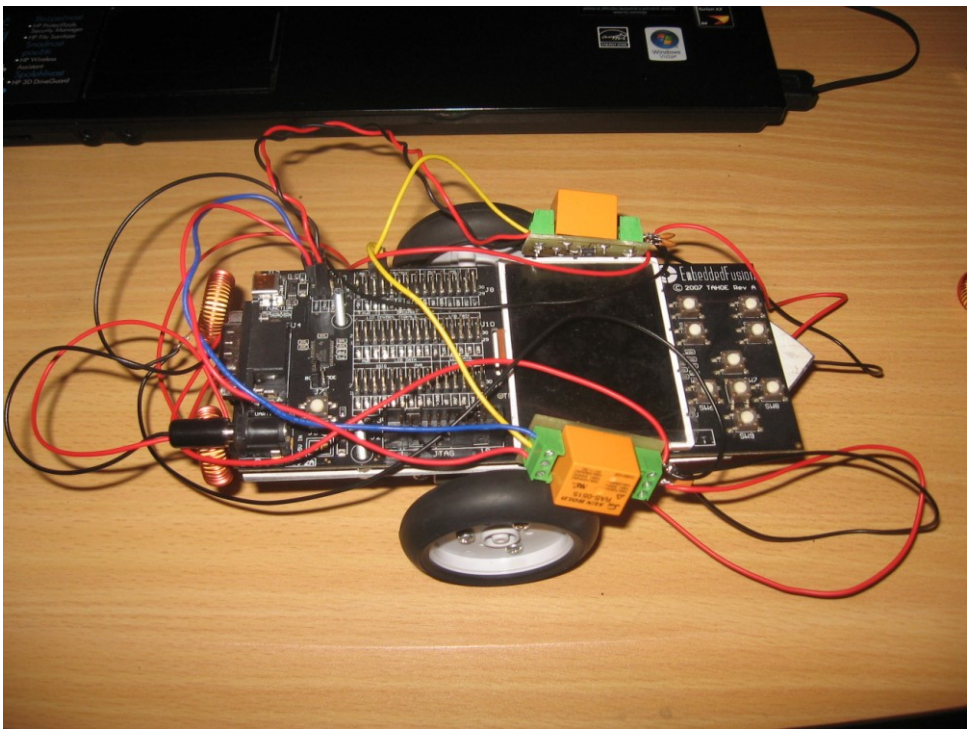
```
MyBitmap.Flush();
```

Smazání vykresleného textu s diplaye Tahoe kitu provedeme následující metodou.

```
MyBitmap.Clear();
```

Funkce motorů byla tímto prověřena ale, ovládání motorů robota o malých rozměrech pomocí velkých relé není příliš efektivním řešením. Když vezmeme v potaz že sepnuté relé má stejný odběr nutný k udržení sepnutí jako běžící motor snižuje nám tato skutečnost radikálně životnost baterií což bylo pro přechod k jinému stylu ovládání hlavním kritériem.

Bylo nutné vyvinout jiný efektivnější systém spouštění motorů, který by byl jak skladnější tak i šetrnější k napájecím článkům.



Obr.5 Robot ovládaný pomocí spínání relé

4.6. Elektronické ovládání motorů

Relé byly pro použití ovládání robota nevhodné bylo nutné vyhledat jiný prostředek, který by umožňoval spouštět programově motory a přitom udržel motory a Tahoe kit galvanicky odděleny. Jako jedno z možných řešení bylo vybráno použití spínacích můstků pro ovládání stejnosměrných motorů.

Spínací můstky slouží jako převodníky řídících signálů na výkonové napájení (buzení) motoru.

Běžné jsou následující můstky:

- *Plný spínací H-můstek* [7] - umožňuje provoz motorů v obou směrech, využívá rychle spínaných MOSFET tranzistorů. Pro ochranu Back-EMF a indukce se připojují diody. Pro řízení bipolárních krokových motorů jsou potřeba dva plné H-můstky.

- *Poloviční H-můstek* [7] - využívá se pro řízení 3fázových a některých stejnosměrných motorů (např. BLDC). Zde se využívá 3 polovičních H-můstků, jeden pro každé vinutí U, V a W. Jedny konce každého vinutí jsou společně spojeny a musí být u řízení spínání tranzistorů zajištěno, aby nebyly zároveň otevřeny oba tranzistory - generování tzv. mrtvého pásma (Dead-Band), tj. překrývání doby PWM impulsů. Pro vysokonapěťové napájení se někdy místo MOSFET tranzistorů využívá IGBT.

Pro naši potřebu budou postačovat dva plné spínací H-můstky, pro každý motor jeden. Vybrán byl integrovaný obvod HT6151A (HT6751B). Jedná se o integrovaný můstek k řízení motorů 2-6V/1000mA, který má odpor horního a dolního spínače 0.4Ohm. Dále potom integrované ochranné diody, tepelnou ochranu. To vše je uloženo v pouzdru SOIC8.

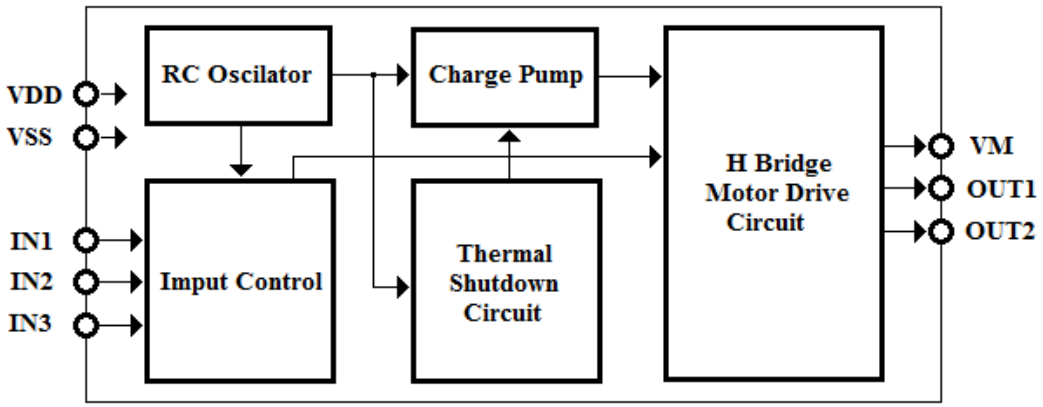
Obvody HT6151A (HT6751B) jsou určeny pro málo výkonové výstupy
pro malé komutátorové motory například v kamerách a v hračkách. Hodí se například pro buzení morů malých robotů. Na plošném spoji zabírají málo místa, protože jsou v pouzdru SO8. Nevýhodou je jednak poměrně malý dovolený výstupní proud do 1000mA a malé napájecí napětí do 6V.

Zvolené zapojení(Obr.13 v příloze na CD) obsahuje 2 obvody stejného typu. Varianty HT6151A (HT6751A a HT6751B) se liší polaritou vstupních signálů a tomu odpovídá i osazení odporů na vstupu tak, aby v klidu byly obvody zaručeně v neaktivním stavu.

Diody D1 a D2 jsou ochranné proti přepólování zdroje. Zenerova dioda D3 slouží jako omezovač napětí, aby se energie vyrobená při doběhu motoru setrvačností nevracela zpět do obvodu můstku.

Napětí na čipu by nemělo překročit za provozu 6V, absolutní maximum je 7V. Pro naše využití jsou tyto podmínky přijatelné, jelikož čtyři články tužkových baterií typu AA nebo AAA mají v součtu 6V.

Obvod HOLTEK HT6151A obsahuje 4 spínací FET tranzistory, řídící logiku včetně nábojové pumpy pro řízení těchto tranzistorů. Při překročení teploty čipu se obvod vypne a po zchladnutí začne opět pracovat.



Obr.6 Vnitřní struktura obvodu HT6151A

Popis rozhraní :

- VDD - kladný pól napájení logického okruhu
- VSS - záporný pól napájení (společný)
- VM - kladný pól napájení motorového okruhu
- OUT1 - silový výstup (kontakt motoru)
- OUT2 - silový výstup (kontakt motoru)
- IN1 - logický vstup 1
- IN2 - logický vstup 2
- IN3 - logický vstup 3

Tabulka (Tab.1) popisuje programového ovládání obvodu s popisem jednotlivých stavů.

HT6151A				
IN1	IN2	IN3	Funkce	Zapnuto
1	0	0	Dopředu	P1 / N2
0	1	0	Dozadu	P2 / N1
1	1	0	Brzda	N1 / N2
0	0	0	Vypnuto	-
1	0	1	-	P2
0	1	1	-	N2
1	1	1	-	N2

Tab.1

Z tabulky programového ovládání obvodu je patrné, že motory lze mimo spuštění v jednom směru spouštět i ve směru opačném a brzdit.

Vytvoření celého modulu s můstky stejně jako modulů pro spínání pomocí relé probíhalo následujícím způsobem.

Obvody pro řízení byly tedy zvoleny a schéma zapojení obvodu nakresleno. Schémata a deska plošného spoje byly navrženy v programu Eagle PCB od firmy CadSoft EAGLE PCB Design Software. Pro vytvoření desky plošného spoje byla vybrána metoda výroby plošných spojů fotocestou. Byla zakoupena kuprextitová deska s měděnou vrstvou potřenou fotocitlivým lakem. Tuto desku bylo nutné upravit na požadovanou velikost a začistit hrany tak aby nebyly ostré. Vytištěné schéma plošného spoje na pauzovací papír se přiloží k této desce na její fotocitlivou vrstvu a uchytí pomocí papírové pásky k rubu desky. Položením desky rubovou stranou na pevnou podložku a přeložíme přes ni skleněnou deskou aby byl papír s vytištěnými cestami přitlačen k fotocitlivé vrstvě plošného spoje. Na skleněnou desku položíme UV lampu kterou 6minut desku osvětlujeme. V místech kde byly vytištěny spoje na puzovacím papíře fotocitlivá vrstva zareaguje a je vidět struktura spojů. Nyní je nutné provézt vyvinutí této desky v roztoku 1,5 hydroxidu sodného (vývojka). Tento postup vyžaduje maximální koncentraci a soustředění. Deska se musí vložit do tohoto roztoku a pozorně sledovat. Postupně se světle žluté cesty začnou vybarvovat do černé. Při zpozorování maximální ostrosti hrany spoje desku okamžitě vyjmout a opláchnout studenou vodou. V případě pozdního zásahu může dojítk rozostření spojů a výsledek nemusí být použitelný pro další postup.

Po opláchnutí studenou vodou desku není žádoucí sušit otřením. Je nutné nechat ji přirozeně zbavit vlhkosti odpařením. Dále desku vložíme do roztoku chloridu železitého pro odstranění mědi, která netvoří spoje. Při průběhu této procedury je nutné desku hlídat a pravidelně sledovat, aby nedošlo k podleptání měděných spojů, to by mělo za následek odpadávání měděných spojů od kuprexitové desky. Po ukončení procedury leptání desku očistíme od zbytku leptacího roztoku studenou vodou, odstraníme lihovou emulzí fotocitlivou vrstvu čímž dostanou spoje lesklost. Následuje vyvrtání děr pro usazení součástek a samotné pájení (Obr.14 v příloze na CD).

Nyní bylo nutné desku oživit (Obr.15 v příloze na CD). Tuto fázi provedeme pomocí laboratorního zdroje stejnosměrného napětí pomalým přiváděním vstupního napětí do obvodu a sledováním chování výstupu. Následně přivádíme vstupní napětí také na vstupy IN1, IN2, IN3 sledujeme chování výstupu. Výsledky všech měření odpovídaly chování obvodu podle tabulky uvedené v předchozím textu (Tab.1). Takto zhotovená deska plošného spoje ovládání motorů byla dále nainstalována na kostru robota a připojena obvodu k modulu Tahoe kitu (Obr. 16 v příloze na CD).

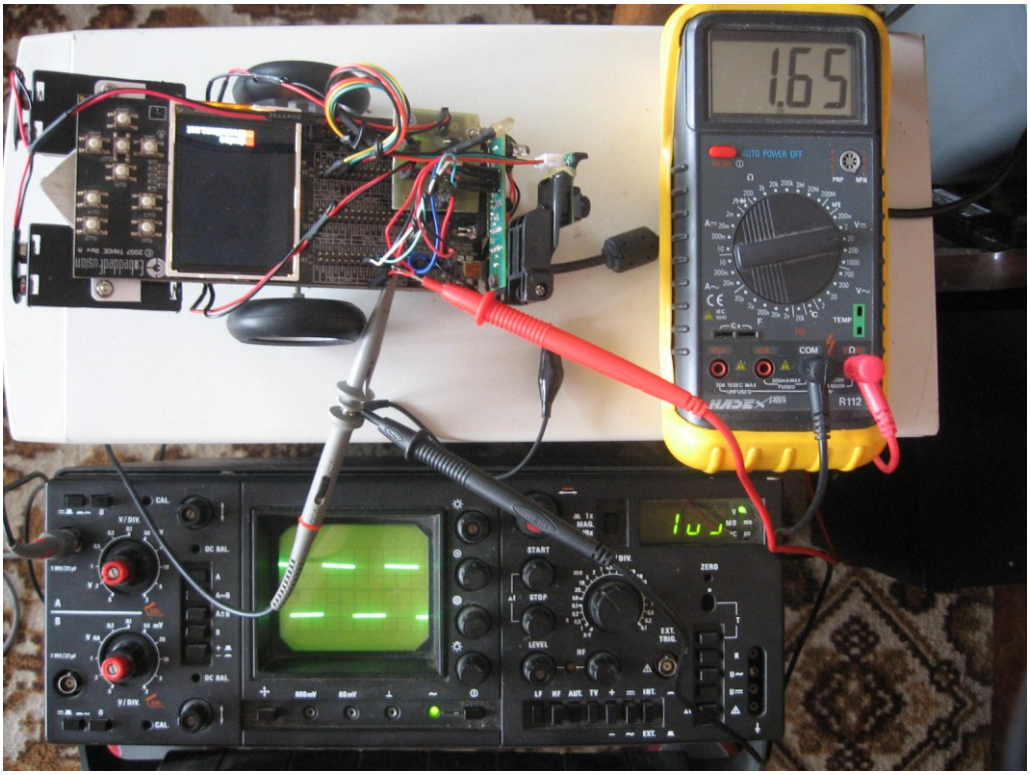
K tomuto účelu byla deska již uzpůsobena v předchozí fázi konstrukce kostry robota, stačilo tedy upevnit desku plošného spoje na již připravený distanční sloupek. Další propojování spočívalo v přivedení napájecího napětí z Tahoe kitu do desky ovládání motorů (k tomuto účelu jsou přímo na desce Tahoe kitu připraveny piny s napájecím napětím 5 a 3,3 voltů).

Dále bylo třeba připojit bateriový systém napájení motorů na k tomu určené vstupní svorky, samotné motory na výstupní svorky obvodu a datové vstupy s výstupními piny Tahoe kitu.

Pomocí následujícího jednoduchého programu byly otestovány výstupní stavy modulu.

```
int delay = 1;
while (true)
{
   Led1_Port.Write(delay);
   Thread.Sleep(1);
   Led1_Port.Write(delay);
   Thread.Sleep(1);
}
```

Jedná se o podmíněný cyklus, jehož podmínka je dána konstantou. Cyklus se bude provádět neustále po celou dobu běhu programu. V tomto cyklu se na výstupní port označený Led1_Port střídavě zapisuje logická jedna a logická nula. Podle nastavení proměnné delay se potom určuje délka držení příslušného pinu v daném stavu. Frekvence změny byla měřena na výstupu pro motory pomocí osciloskopu. Průběh měření je dobře patrný z obrázku č. 24. Měření proběhlo podle očekávání a modul byl tedy schopen ostrého provozu. Po připojení motorů bylo otestováno rozbíhání jednotlivě i souběžně s neočekávaným výsledkem.



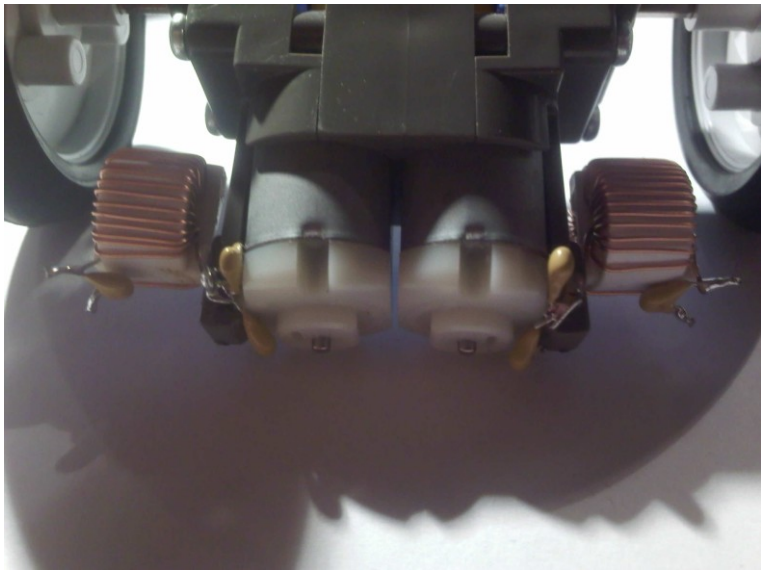
Obr. 7. měření výstupu motorů generováním různých frekvencí změny stavu na výstupu

Vlivem elektromagnetického rušení během testu docházelo restartování Tahoe kitu. Motory se rozeběhly, ale bylo patrné podle problikávání displaye, že vytvářejí silné elektromagnetické rušení, které narušuje správnou funkci Tahoe kitu. Po pár vteřinách chodu motorů se Tahoe kit restartoval a po naběhnutí zavedeného programu na testování motorů se celá situace opakovala.

Elektromagnetické rušení vzniká vlivem komutace - přechodů komutátorových kartáčků mezi jednotlivými články komutátoru.

Komutátor[8] jako poměrně složitá mechanická součást přenášející velké proudy bývá zdrojem nežádoucích mechanických poruch či elektromagnetického rušení, jež vzniká vlivem nežádoucího jiskření. To nastává především vlivem vzniku vzduchových mezer mezi sběračem proudu a povrchem komutátoru. Vzhledem k tomuto jiskření mezi kartáči a komutátorem nejsou stroje vybavené komutátory vhodné do prostředí s možností výskytu hořlavých či výbušných plynů.

Elektrické motorky pohonu jsou tedy zdrojem rušení, proto je potřeba na těchto jednoduchých stejnosměrných motorcích provést odrušení. Vhodný kondensátor připájíme rovnou na vývody motorku. Většinou vyhoví kondensátory o kapacitě 0,1 mikro-Faradu, keramické, ploché. Vhodné je přidat i tlumivky. Což jsou cívky navinuté z lakovaného měděného drátu o průměru 0,3 až 0,5mm na feritová jádra o kruhovém průřezu. Vývody se snažíme mít co nejkratší. Různé způsoby zapojení kondensátorů na kontaktech motorků ukazuje obrázek č.25 v příloze na CD. Tlumivky můžete přidat k jakémukoliv způsobu zapojení. Po instalaci tohoto jednoduchého odrušovacího obvodu se již další restartování Tahou kitu neprovedlo a rušení bylo eliminováno na minimum. Další testování pohybu pomocí motorů již probýhalo dle předpokladů. Schéma zapojení je uvedeno v příloze na CD jako Obr.17.



Obr.8 nainstalované odrušení na motorech robota

5. KONSTRUKCE ZÁKLADNÍHO ROZEZNÁVACÍHO ROZHRANÍ

Jako základní systém pro orientaci robota v prostoru byl zvolen způsob detekce překážek pomocí IR čidel. K tomu byl zakoupen IR senzor[9] od firmy SnailInstruments ProxDet01.

Senzor zjistí, zda je před robotem překážka a na které je straně. Podle strany změní stav na příslušném výstupním pinu a tím dá možnost řídícímu programu v Tahoe kitu reagovat a provézt korekci směru.

Analýzou dostupných informací[9] o tomto senzoru bylo zjištěno, že velmi jednoduchý detektor překážek lze sestavit ze dvou infračerveně vyzařujících LED diod (zde nazývaných vysílací) a přijímače pro infračervené dálkové ovládání typu SFH5110. Jedna vysílací LED ozařuje prostor vpravo a uprostřed, druhá vlevo a uprostřed, takže společným vyhodnocením odrazů od jedné nebo druhé LED umožní přibližnou lokalizaci polohy překážky. Protože přijímač vyhodnocuje pouze intenzitu odraženého záření, je výsledek samozřejmě závislý nejenom na vzdálenosti, ale i na velikosti a odrazivosti překážky pro blízké infračervené záření. Nemůžeme tedy, na rozdíl od senzorů skutečně dálkoměrných, určit vzdálenost přesně,
ale jen zda překážka je nebo není přítomna, a pokud ano, tak zda je spíše vpravo, vlevo či uprostřed. Tato omezení jsou však bohatě vyvážena jednoduchou konstrukcí a nízkou cenou. Přijímač SFH5110-38 reaguje na dopadající infračervené záření, modulované kmitočtem 38 kHz (pro úplnost – existují i varianty, určené pro modulační kmitočty 36 kHz a 40 kHz). Pokud přijímač detekuje záření správné frekvence a dostatečné intenzity, uvede svůj výstup OUT do nízkého stavu. Na infračervené

záření bez modulace přijímač nereaguje, takže není rušen žárovkami nebo zářivkami umělého osvětlení. Samozřejmě odolnost vůči rušení má své meze, na přímém slunečním světle dojde k zahlcení vstupních obvodů. Pro venkovní použití se tedy tato součástka příliš nehodí. Přijímač SFH5110 obsahuje, vzhledem ke svému původnímu určení (pro dálkové ovládání spotřební elektroniky), obvod automatického vyrovnávání citlivosti – pokud přijme intenzivní signál, snižuje svoji citlivost, naopak při příjmu slabého nebo dokonce žádného signálu citlivost zvyšuje. Ovšem změna citlivosti není okamžitá, ale nastává se zpožděním několika milisekund. Takovéto chování je pro naše účely poněkud nevýhodné – signál s frekvencí 38 kHz nemůžeme vysílat trvale, ale je

třeba ho pravidelně přerušovat a nechat přijímač vždy několik milisekund odpočinout, aby se obnovila jeho citlivost.

K řízení dvou vysílacích LED a zároveň také k vyhodnocování odraženého signálu nám dobře poslouží mikrokontrolér PICAXE-08M2. Jak je vidět z předchozího textu, generování správného signálu pro vysílací infračervenou LED je klíčem k úspěchu. Ta se musí 38 000 krát za sekundu rozsvítit a zase zhasnout, toto „blikání“ by mělo trvat asi 0,5 ms a potom musí LED zůstat přibližně 5 ms zhasnutá. A protože máme dvě vysílací LED, které ozařují prostor vpravo a vlevo, budou se po 5 ms střídat. Mikrokontrolér PICAXE používá k vytvoření signálu o kmitočtu 38 kHz příkaz PWMOUT. Protože tento příkaz je u použitého

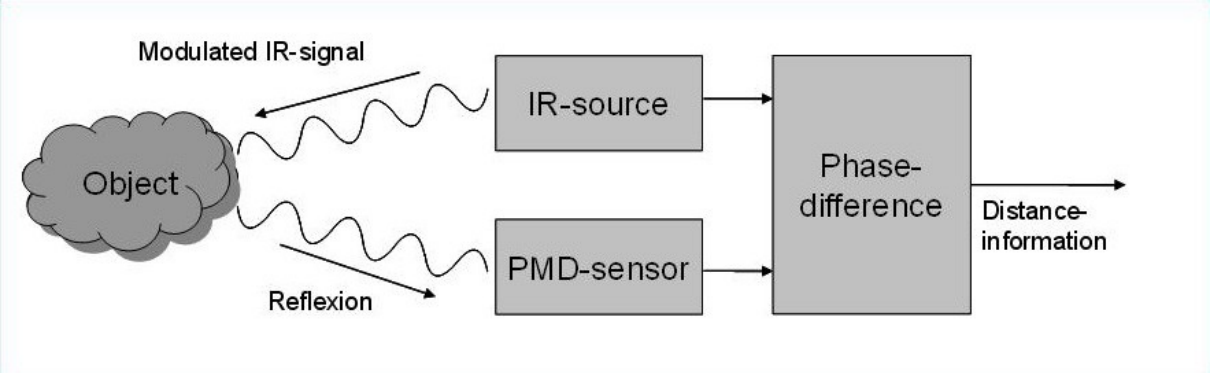
mikrokontroléru PICAXE-08M2 omezen pouze na pin 2, jsou vysílací diody zapojeny antiparalelně. D1 svítí, pokud je výstup 1 na vysoké úrovni a výstup 2 na nízké, zatímco D2 svítí, pokud jsou logické úrovně opačné. Na vstupu 3 snímá PICAXE odpověď přijímače a výsledek signalizuje nadřízenému systému logickými stavy na výstupech 0 a 4. Program v PICAXE-08M2 využívá ještě další trik – regulace intenzity záření změnou šířky pulsu. Intenzita střídavého signálu s kmitočtem 38 kHz je nejvyšší při činiteli plnění 50%, pokud snížíme činitel plnění třeba na 20% (při zachování frekvence 38 kHz), důsledek je stejný, jako kdybychom zvětšili hodnotu odporu R1 a tím zeslabili intenzitu záření LED. Tímto způsobem můžeme do určité míry měnit dosah detektoru. V podrobně komentovaném výpisu programu snad jediné místo zasluhuje podrobnější vysvětlení – činitel plnění 75% pro pravou diodu. Jak jsme konstatovali, tato dioda svítí při vysoké (L) úrovni na vývodu 1 (což zabezpečuje příkaz high PWMright) a nízké úrovni na vývodu 2. Protože činitel plnění v příkazu pwmout se vztahuje vždy k aktivní vysoké úrovni, bude činitel plnění pro aktivní nízkou úroveň doplněk do 100%, čili v našem případě 25%. Správné parametry příkazu pwmout nám pomůže spočítat wizard z vývojového prostředí kontrolérů PICAXE. Součástky C3, C4 a R6 slouží k filtraci napájení pro infračervený přijímač, R2 a R4 chrání PICAXE při chybném propojení s nadřízeným procesorem.

Snímací IR Senzor byl umístěn na přední část robota, tak aby nebránil v přístupu k žádnému z portů, ale zároveň tak, aby měl co nejvíce prostoru pro detekci objektů. Napájen je z vývojové desky Tahoe kitu stejně jako modul řízení motorů a logická část. Testování správné funkce IR čidla bylo provedeno měřením, pomocí dvou digitálních multimetrů (Měření reprezentují obrázky 18-27 v příloze na CD). Multimetry byly připojeny na levý a pravý výstup detektoru a signalizovaly zakrytí levé, pravé nebo obou IR diod uváděním výstupních pinů do příslušných stavů (volný prostor - Low - 0V, zakrytí diody - High - 5V).

Obr.9 umístění IR senzoru na přední části robota

Bylo zjištěno že existuje metoda daleko přesnějšího využití IR záření. Bohužel nebylo nalezeno vhodné zařízení, které by se dalo pro tuto metodu v případě konstrukce našeho robota využít . Jedná se o metodu Time-Of-Flight Sensors, dále jen TOF.

TOF senzory pracují na principu měření času vyslaného laserového paprsku směrem k pozorovanému objektu a jeho cesty zpět do detektoru, čímž se zjistí vzdálenost objektu (Obr. 51). Jelikož TOF senzory používají laserový paprsek, nazývají se také někdy LIDAR (Light Detection And Ranging) nebo LADAR (laserové lokátory).



Obr. 10 Princip TOF senzorů [16]

TOF senzory [15] většinou vysílají pouze jeden laserový paprsek. To znamená, že z povrchu součásti získáme informaci v podobě bodu. V praxi ovšem potřebujeme znát více informací. Proto je paprsek hnán soustavou zrcátek po povrchu objektu a tím získáme data v podobě vektoru.

K pohybu zrcátek se používá krokový motor nebo rotační či oscilační zrcátka pro automatické skenování. Udávaná přesnost TOF senzorů je 5 - 10 mm. Mezi výrobce těchto senzorů patří SICK, Mensi, DeltaSphere nebo Acuity.

6. VYTVOŘENÍ PROGRAMOVÝCH OVLÁDACÍCH PRVKŮ PRO POHYB ROBOTA POMOCÍ IR ČIDEL

V této kapitole se budu věnovat programovému ovládání robota pomocí IR čidel. V první fázi byly vytvořeny inicializační struktury pro jednotlivé piny a porty v závislosti na jejich určení. Bylo potřeba vymezit dva piny pro logické vstupy IR čidel a dva piny určené k ovládání motorů.
V následující struktuře byly inicializovány potřebné piny.

GPIO01 a GPIO02 jsou vyhrazeny pro funkci dvoubarevné diody, která indikuje průběh hlavního vlákna pro kontrolu uváznutí nebo chyby běhu programu.

```
static Cpu.Pin Led_L1 = Meridian.Pins.GPIO1;
static Cpu.Pin Led_L2 = Meridian.Pins.GPIO2;
```

GPIO03 a GPIO04 jsou vyhrazeny pro práci s motory a jejich spínání a vypínání pomocí modulu můstku. Jak již bylo uvedeno modul můstku dokáže s motory pracovat více způsoby. Nám bude zatím pro práci s IR čidly postačovat ovládání směru v před a stop, tedy bude postačovat pro každý motor jeden pin.

```
static Cpu.Pin Motor_Rigth = Meridian.Pins.GPIO3;
static Cpu.Pin Motor_Left = Meridian.Pins.GPIO4;
```

GPIO05 a GPIO06 jsou vyhrazeny pro práci s IR čidly. Výstupní úrovně IR modulu budou analyzovány podle programu v Tahoe kitu a po vyhodnocení bude upraven chod motorů.

```
static Cpu.Pin IR_Left_port = Meridian.Pins.GPIO5;
static Cpu.Pin IR_Right_port = Meridian.Pins.GPIO6;
```

Předchozí nastavení bylo pro všechny vstupně výstupní piny totožné. Následující popis se bude pro jednotlivé piny lišit. Piny určené pro ovládání motorů a indikace led diodou budou nastaveny jako výstupní a porty určené pro přijímání výsledku analýzy z IR čidel jako vstupní.

Inicializace výstupních portů pro indikaci lediodou a řízení motorů.

```
public static OutputPort Led1_Port = new OutputPort(Led_L1, false);
public static OutputPort Led2_Port = new OutputPort(Led_L2, false);
public static OutputPort Motor_Rigth_Port = new OutputPort(Motor_Rigth, false);
public static OutputPort Motor_Left_Port = new OutputPort(Motor_Left, false);
```

Inicializace vstupních portů zachytávajících informaci IR čidel.

```
private static InterruptPort IR_Left = new InterruptPort(IR_Left_port, false,
Port.ResistorMode.Disabled, Port.InterruptMode.InterruptEdgeBoth);
private static InterruptPort IR_Right = new InterruptPort(IR_Right_port, false,
Port.ResistorMode.Disabled, Port.InterruptMode.InterruptEdgeBoth);
```

Registrace událostí, které budou vyvolány v případě změny stavu vstupních portů na které je připojeno IR čidlo.

```
IR_Left.OnInterrupt += new NativeEventHandler(port_OnInterrupt_IRLeft);
IR_Right.OnInterrupt += new NativeEventHandler(port_OnInterrupt_IRRight);
```

Metody volané vyvoláním událostí změny stavu IR čidla. V těchto metodách je umístěno volání výkonného kódu, který řídí chod motorů. Následně podle zjištění překážky proběhla příslušná reakce.

```
private void port_OnInterrupt_IRLeft(uint port, uint state, TimeSpan time) {}
private void port_OnInterrupt_IRRight(uint port, uint state, TimeSpan time) {}
```

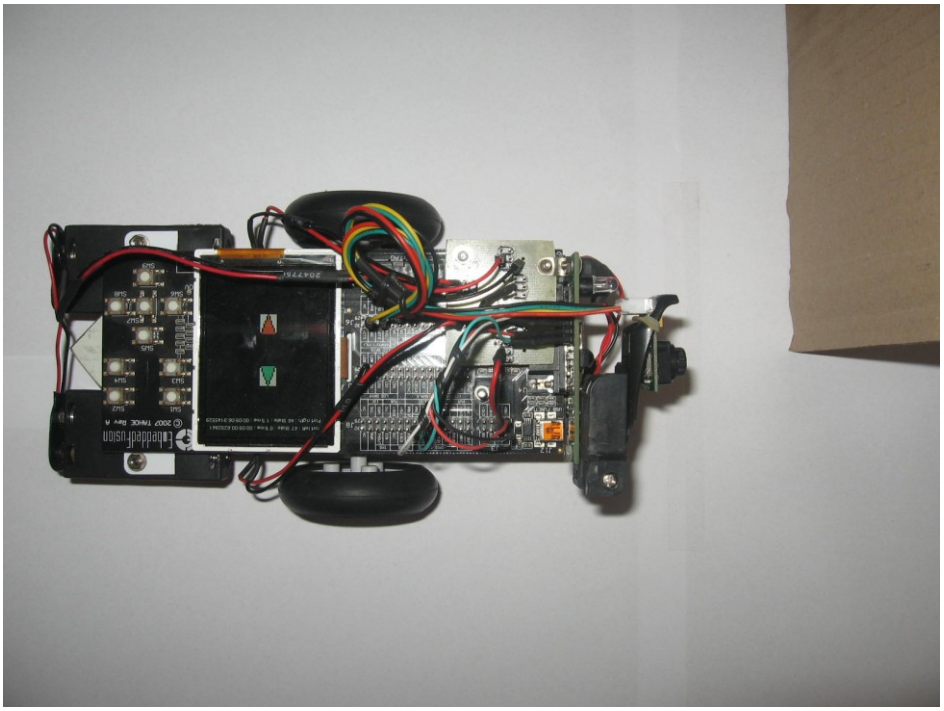
Popisy parametrů jednotlivých funkcí a konstruktorů byly popsány v úvodní kapitole, proto se jimi v tuto chvíli již nebudeme zabývat.

Výkonná část kódu je napsána tak. že v případě detekce objektu IR čidlem na levé straně je zastaven pravý motor a levý zůstává v chodu, čímž je docíleno vyhnutí se překážce zatočením vpravo. Pokud již není objekt IR čidlem detekován je rozběhnu pravý motor a robot jede rovně. Obdobně pokud je detekována překážka na pravé straně je zastaven levý motor čímž dojde k vyhnutí se překážce vlevo. Jakmile není objekt detekován je opět levý motor spuštěn.

Maximální vzdálenost objektu na detekci čidlem je 13cm při plně nabitých bateriích. Se snižující se kapacitou baterií se také snižuje citlivost čidel. Výkonný kód k této části je součástí přílohy na CD k této práci (Příloha.15).

Během běhu programu není indikován jen stav hlavního vlákna blikáním diody ,ale je také vypisován aktuální stav detekce IR čidla na display. Tato informace má následující strukturu. Je vypisováno o který port se jedná, jeho číslo vzhledem k vnitřní struktuře Tahoe kitu, do jaké stavu se výstup čidla dostal (1 - High - 5V, 0 - Low-0V) a doba trvání přechodu z původní úrovně do úrovně nové . Pro lepší pochopení aktuálního stavu čidel byla přidána indikace stavu čidla pomocí barevných šipek. Pro každou stranu je vykreslována zelená šipka ve směru detekce v případě, že je prostor před robotem volný a červená šipka v případě, že robot narazil na překážku. Celý postup testování je patrný z obrazků č. 34 - 37 na CD v příloze k této diplomové práci. (Příloha.15)

V tuto chvíli je robot schopen pohybu podle IR detektoru.



Obr.11 Test modulu IR čidla pro levou stranu

7. SEZNÁMENÍ SE S POHYBEM V PROSTORU

Vyvíjený robot se pohybuje v prostoru za pomocí dvou stejnosměrných motorů, které při správném naprogramování řídící jednotky dokáží robota vézt rovně, zatáčet a otáčet v kruhu napravo nebo nalevo podle toho jaký směr program zvolí. Při vložení programu může robot absolvovat jakoukoliv trasu mezi překážkami aniž by se překážek dotknul. Vše záleží na správném odhadu vzdáleností programátora který program vytváří. Způsob práce takového robota by, ale nebyla efektivní pokud by projížděl stále stejnou trasu bez možnosti přizpůsobovat se změnám v prostoru. Programátor by musel stále měnit a vkládat program podle toho jaké změny by proběhly v trase jízdy robota, což by bylo velice nepraktické a časově náročné, navíc pokud by se objekty na trase měnili v reálném čase , tak by programátor nikdy kód nedopsal, protože během té doby než by vyvinul nový kód programu by se trasa změnila a on by mohl začít psát program zase od začátku. Nejjednodušším řešením by bylo dát robotu možnost, aby se v prostoru mohl orientovat. Toto povědomí o okolním světě je u našeho robota realizováno modulem s IR čidly. Tento jednoduchý prvek dává robotu sice jen minimální možnost dozvědět se o stavu světa kolem něj víc a případně na tento stav reagovat, ale i to stačí na to aby se robot mohl pohybovat v prostoru a vyhýbat se překážkám které by vyskytli v jeho dráze.

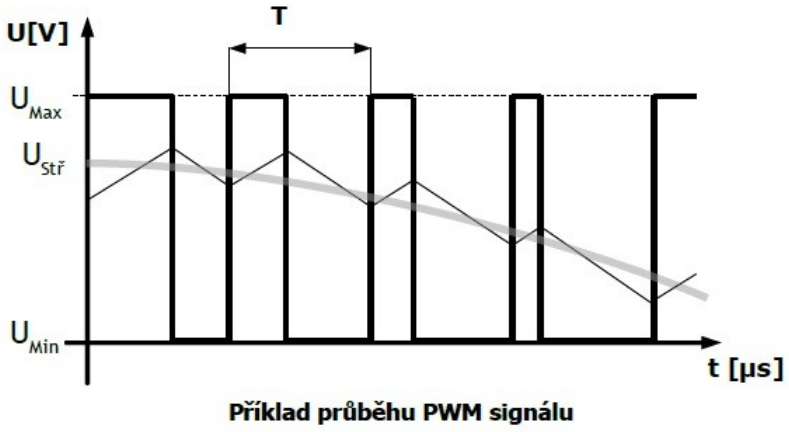
Nyní jsme ve vývoji dospěli do fáze kdy máme funkční pohonný systém který, se dá ovládat a zajistit pohyb robota. Máme také jednoduchou detekci okolního světa pomocí IR čidel, které nám zajišťují ochranu proti případné kolizi s objekty v trase.

Program pro toto testování byl sestaven, tak jak bylo popsáno v předchozí kapitole, ale i to by mělo stačit, aby robot absolvoval jakoukoliv trasu. Nejhorší možný výsledek který by pro robota mohl nastat je ten, že by obě čidla detekovala překážku. V takovém případě by robot podle výše popsaného programu zastavil a zůstal stát, což by zabránilo jeho poškození.

Během testování pohybu robota došlo ke zjištění, že pohyb robota byl i přes minimální možné nastavení převodovky příliš rychlý. Při výskytu překážky, pokud byl robot v jízdě v přímém směru a narazil na překážku, vypínal motor podle strany na které byla překážka detekována. Setrvačnost točení vypnutého motoru byla příliš vysoká a tak došlo k tomu , že robot překážku nestačil minout. Točení v případě doběhu vypnutého motoru bylo málo razantní. Detekce překážky pomocí IR čidel sebou nese také nevýhody týkající se vlastního IR záření.

Modulované IR záření se odráží od různých povrchů s různou intenzitou, z čehož vyplynulo a při testování se ukázalo být značnou nevýhodou, že se vzdálenost jednotlivých objektů liší v závislosti na jejich povrchu - IR čidlo detekuje odraz záření až při větším přiblížení k takovému objektu. Bylo tedy nutné určitým způsobem eliminovat chyby vznikající vysokou rychlostí jízdy robota. Jelikož sestavení převodovky bylo provedeno podle přiloženého manuálu na minimální možnou konfiguraci co se týká otáček výstupní hřídele a napájecí napětí motorů se snížit vzhledem ke konstrukci ovládacího prvku motorů nedalo. To by bylo možné kdyby se místo modulu s H-můstky použila konstrukce D/A převodníku který by podle stavu na vstupech určenými logickou hodnotou upravil hladinu výstupního napětí. Bylo nutné vyvinout, nebo použít nějaký jiný způsob snížení otáček stejnosměrného motoru.

Nejpoužívanějším způsobem řízení motorů je pulsně-šířková modulace výstupního signálu. Pulsně-šířková modulace[10], neboli PWM (Pulse Width Modulation) je diskrétní modulace pro přenos analogového signálu pomocí dvouhodnotového signálu. Jako dvouhodnotová veličina může být použito například napětí, proud, nebo světelný tok. Signál je přenášen pomocí střídy. Pro demodulaci takového signálu pak stačí dolnofrekvenční propust. Vzhledem ke svým vlastnostem je pulsně šířková modulace často využívána ve výkonové elektronice pro řízení velikosti napětí nebo proudu. V našem případě se bude jednat o řízení velikosti napětí. Kombinace PWM[13] modulátoru a dolnofrekvenční propusti bývá rovněž využívána jako levná náhrada D/A převodníku. Přenosový signál, který nese informaci o přenášené hodnotě může nabývat hodnot zapnuto/vypnuto tj. log. 1 / log.0. Hodnota přenášeného signálu je v přenosu "zakódována" jako poměr mezi stavy zapnuto/vypnuto. Tomuto poměru se říká střída. Cyklus, kdy dojde k přenosu jedné střídy se říká perioda. Časové hodnoty střídy se pohybují v milisekundách pro přesnější řízení. Perioda je vždy součtem doby zapnuto a vypnuto.



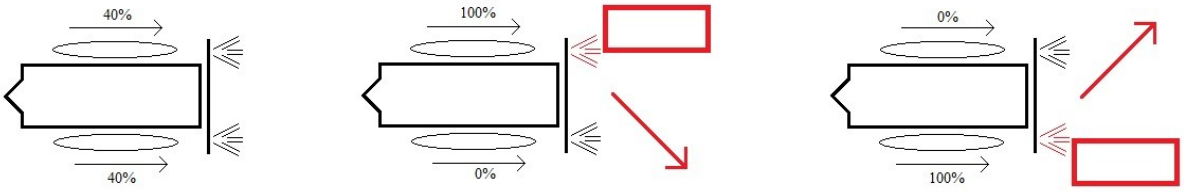
Obr.12 Plusně-šířková modulace

Vzhledem ke stávající koncepci robota bylo tedy nutné uchýlit se k výše zmíněnému ovládání motorů.

Do této chvíle byla celá pohybová koncepce robota postavena na jednom procesu, který obstarával všechny funkce tzn.: staral se jak o chod motorů (stávající situace poskytovala pouze dva stavy chodu motorů což znamená, motor zapnutý (plné otáčky) motor vypnutý (bez napájení, nulové otáčky)) tak o výpis informací na display (stav čidel - čistý prostor před, robotem zelená šipka, objekt zaregistrován, reakce na překážku, červená šipka a to pro obě strany). Dále se staral o reakci na objekty zjištěné IR čidly. Podle výše zjištěných informací o použití pulsně-šířkové modulace pro ovládání stejnosměrných motorů bylo nutné rozdělit tyto funkce mezi minimálně 3 procesy. Jeden proces pro ovládání výkonu levého motoru, druhý proces pro ovládání výkonu pravého motoru a poslední proces pro logiku chování robota.

Proces pro ovládání motoru má za úkol měnit pulzně-šířkovou modulaci pro motor a tím měnit jeho výkon. Hlavní proces ovládá tento vedlejší proces jednou globální proměnnou typu int, která může obsahovat hodnoty 0-100. Tyto hodnoty odpovídají 0 - 100% výkonu motoru.

V programu tyto hodnoty udávají jak dlouho je během 100 milisekundového intervalu na výstupu pro ovládání motoru na vysoká úroveň. Což v praxi znamená že pokud je ve výše zmíněné proměnné hodnota 100 , je po dobu 100 milisekund výstupní pin Tahoe kitu určený pro ovládání motoru ve vysoké úrovni a tedy motor jede na 100% výkonu. Pokud je v proměnné uvedena hodnota například 50, tak to znamená že výstup určení pro ovládání motoru na Tahoe kitu bude 50 milisekund ve vysoké úrovni (5V) a 50 milisekund v úrovni nízké (0V), tedy motor jede na 50% výkonu atd. podle zvoleného čísla v proměnné. Kódy procesů určené pro levý a pravý motor jsou identické, tedy není potřebovat proces pro motor pravý a pro motor levý vysvětlovat zvlášť. V rámci testování byl rozdělen výkon motorů po 10ti procentech a otestován nejpříhodnější stav pro pohyb v před a zatáčení při objíždění. Závěr, který vyplynul z testování byl takový, že nejvýhodnější pro pohyb v před je výkon 40% pro oba motory. Dále potom 0% pro pravý motor a 100% pro levý motor při objíždění překážky vlevo a 0% pro levý a 100% pro pravý pokud se překážka vyskytuje na pravé straně. Informace o chodu motorů byly vypisovány na diplay pro každý motor zvlášť.



Obr.13 Obrázek průjezdu robota kolem překážek s popisem výkonu motorů.

Dále bylo nutné zařídit spouštění a vypínání chodu robota. Bylo nepříjemné pokud se při natažení odlaďovaného programu do Tahoe kitu robot hned rozjel i se zapojeným USB kabelem.

Pro tuto úlohu jsem použil tlačítka vlastní konstrukce připojená k pinům Tahoe kitu. Zvolené piny byly nastaveny podle výše zmíněných postupů jako vstupní, jen s tím rozdílem , že jelikož se jedná o tlačítka byly změněny některé paramery v konstruktoru třídy InteruptPort a to následujícím způdobem.

InterruptPort TlStart = new InterruptPort(TlacitkoStart, true, Port.ResistorMode.PullUp, Port.InterruptMode.InterruptEdgeLow);

V prvním parametru zůstává zvolený pin na který připojíme jeden kontakt tlačítka, to se vzhledem k předchozím informacím nemění. Jako druhý parametr jak bylo popsáno výše je takzvaný "Glitch Filter" - filtr zákmitu při sepnutí tlačítka. Tento parametr jsme v předchozích případech nechávali nastaven na hodnotu false, tedy filtr vypnutý a to protože se jednalo o změnu stavu vstupního pinu nemechanickou cestou, čili nemohlo k zákmitům dojít. Tentokrát jej nastavíme na hodnotu true, čili necháme filtr zákmitů zapnutý.
Další důležitou věcí je rozhodnout z jakého do jakého stavu budete stiskem tlačítka přecházet, potažmo k jakému pinu připojíme druhý vodič tlačítka. Máme dvě volby.

Buďto budeme reagovat na přechod z úrovně Low (0V) do úrovně Higth (5V). V tom případě připojíme druhý kontakt tlačítka k hodnotě 5V, tedy napájecímu napětí Tahoe kitu. Vývody s napájecím napětím jsou vyvedeny přímo na Tahoe kitu. Tato varianta byla vyzkoušena, ale nedoporučuji ji, vzhledem k tomu že toto nastavení bylo aktivní až při běhu programu. Pokud se Tahoe kit nacházel ve stavu kdy nebyl k dispozici program a stisklo se takto zapojené tlačítko docházelo ke zkratu a restartu Tahoe kitu, jelikož všechny porty Tahoe kitu jsou v tomto stavu nastaveny na nízkou úroveň (Low - 0V) a stiskem tlačítka se na tyto piny přivádělo napájecí napětí.

Pokud by jsme se i přes tuto nevýhodu rozhodli reagovat na náběžnou hranu, tedy přechod Low (5V) na Higth (0V) provedeme nastavení posledních dvou parametrů následujícím způsobem. Předposlední. parametr nastavíme Port.ResistorMode.PullDown a poslední parametr Port.InterruptMode.InterruptEdgeHigth.

Nebo můžeme reagovat na přechod z úrovně vysoké (Higth - 5V) do úrovně nízké (Low - 0V). V tomto případě připojíme druhý konec spínače na nulový vývod napájecího napětí, který je také vyvedený pro tyto případy na desce Tahoe kitu. Veškeré starosti týkající se výše zmíněného problému odpadají. Pokud by v Tahoe kitu nebyl zaveden program a sepnuli jsme tlačítko nestane se nic závažného. Piny jsou inicializovány na úroveň nízkou a sepnutím tlačítka se na ně přivede také nízká úroveň, tak že ke zkratu nemůže nedojít.

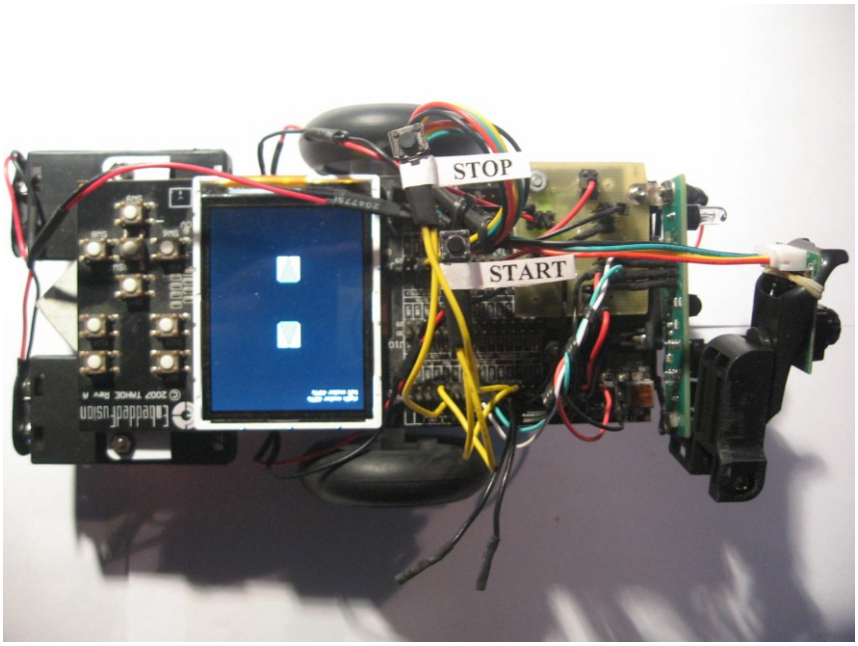
Toto nastavení jsem použil pro funkci tlačítka start a tlačítka stop. Nastavení předposledního a posledního parametru pro inicializaci portu k tomuto použití je následující. Předposlední parametr nastavíme na hodnotu Port.ResistorMode.PullUp, poslední parametr nastavíme na hodnotu Port.InterruptMode.InterruptEdgeLow.

Dále bude popsána programová funkcionalita tlačítek, před tím je, ale nutné upozornit na ještě na některá programová nastavení pro lepší pochopení popisovaných funkcí tlačítek.

Při detekci objektu IR čidly je použito dvou booleovských proměnných, které jsou nastavovány podle stavu čidel (statusL, statusR). Tyto proměnné udávají jak jsou nastaveny rychlostní parametry motorů (statusL = false a statusR = false - oba motory plný výkon (pro pohyb v před přímí 40%), statusL = true a statusR = true - oba motory nulový výkon (0%), statusL = true a statusR = false - levý motor plný výkon (100%) - pravý nulový výkon(0%) , statusL = false a statusR = true - levý motory nulový výkon(0%) - pravý motor plný výkon(100%)). Dále potom nastavujeme booleovskou proměnnou která udává jestli bude na událost vyvolanou změnou stavu na IR čidlech reagováno nebo bude ignorována (paused = true - ignorvat události, paused = flase - reagovat na události).

Tedy při použití tlačítka STOP byly nastaveny proměnné "statusL" na true a "statusR" na true, jako kdyby obě čidla detekovali překážku, což znamená že oba motory byly zastaveny. Dále potom byla proměnná "paused" nastavena na true čímž se ignoruje vyvolání událostí od IR čidel. Tímto byl celý robot zastaven a na display bylo vypsáno hlášení o zastavení systému.

Při použití tlačítka START byly proměnné nastaveny přesně na opačné hodnoty čímž se rozběhly motory na maximální výkon pro pohyb v před (40%) a bylo povoleno vyvolání událostí od IR čidel.



Obr.14 Obrázek Tahoe kitu s tlačítky start a stop.

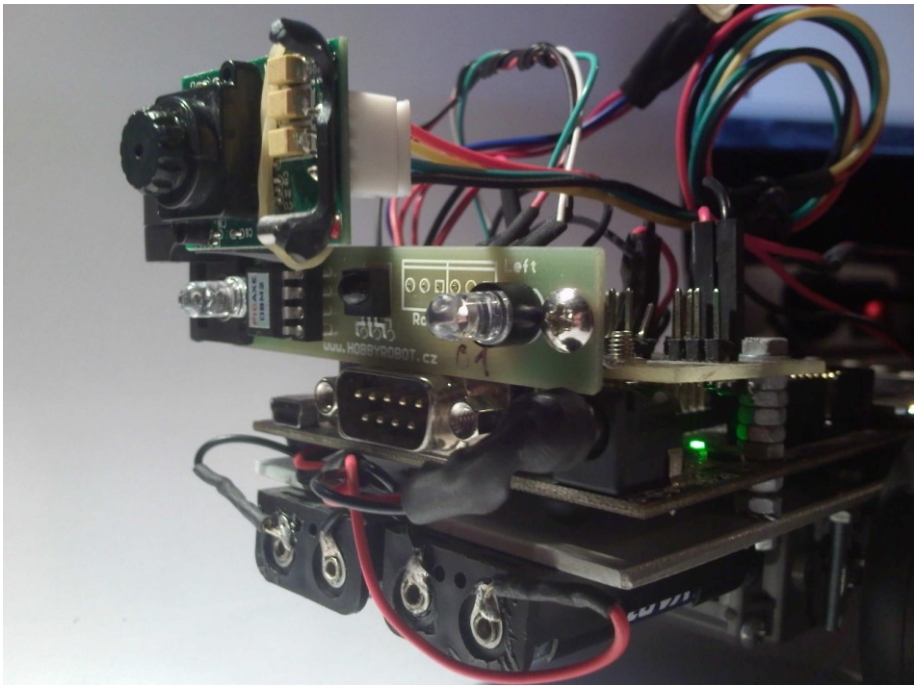
Nynější vývojový stav robota byl následující. Robot byl schopen reagovat na překážky a celkem efektivně se jim vyhýbat. Také bylo otestováno fungování robota ve venkovních podmínkách, což dopadlo podle očekávání. IR záření produkované sluncem způsobovalo značnou nepřesnost při detekci objektů pomocí IR čidel. Tento problém již byl popsán výše v kapitole zabývající se IR čidly.

8. POUŽITÁ KAMERA A KONSTRUKCE DRŽÁKU KAMERY

Jako další stupeň detekce objektů v prostoru byla zvolena kamera. Pomocí kamery je vytvořen snímek a tento snímek je dále pomocí hranového detektoru zanalyzován. Výsledek této analýzy potom určí směr pohybu. K dispozici pro konstrukci robota byla dána kamera od firmy COMedia Ltd, C328R Jpeg Compression VGA Camera Module. Kamery (Obr.29 v příloze na CD) tohoto výrobce mají tu výhodu že je výsledný obraz přenášen pomocí protokolu RS232, tedy sériovým přenosem ke kterému stačí dva vodiče pro datový přenos a dva vodiče pro napájení kamery. Většina kamer má v dnešní době minimálně dvacet výstupních vodičů, které musí být zpravovány. Tato skutečnost je nemalou překážkou při práci s takovou kamerou a vyžaduje značnou programovou administraci. Výhody jsou nízké pořizovací náklady při koupi kamery, nízké nároky na napájecí napětí, možnost volby rozlišení obrazu mezi VGA/CIF/SIF/QCIF/160x128/80x64, JPEG CODEC pro různá nastavení rozlišení, automatická detekce přenosové rychlosti datového toku sériové komunikace. Další velkou výhodou je velikost samotného modulu, který má rozměry 20x28 mm. Díky těmto rozměrům nebylo složité pro kameru sestrojit systém uchycení.

V původním konceptu bylo počítáno s pohyblivou kamerou, která by se natáčela v úhlu 180 stupňů z jedné strany robota na druhou pomocí servomotoru. Díky tomuto systému mělo být dosaženo získání více snímků, které se měly spojit v jeden panoramatický obraz objektů před kamerou. Takovýto obraz měl být dále analyzován a měl být zvolen směr pohybu k vybraným objektům.

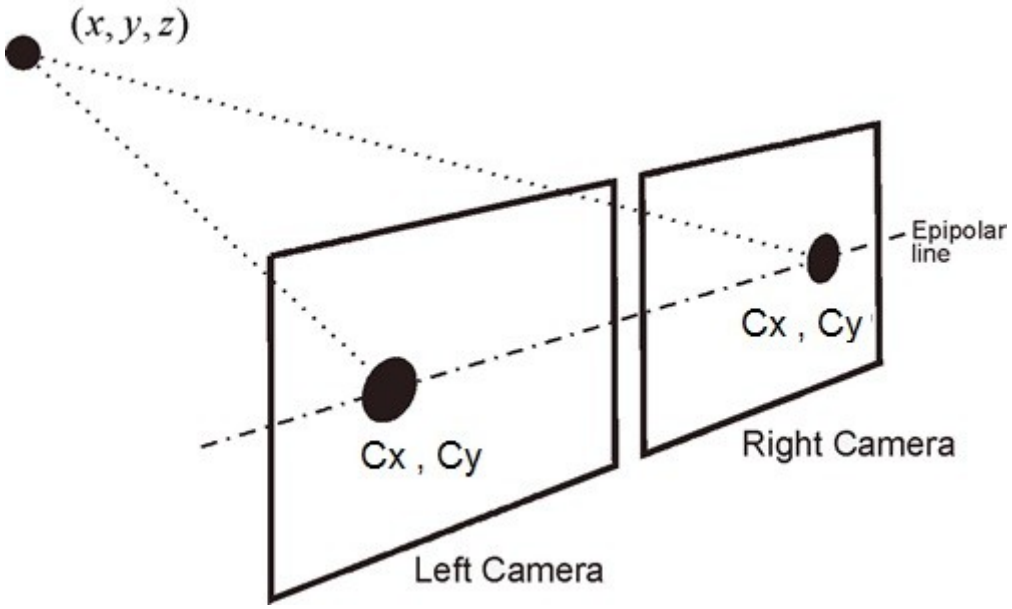
Vzhledem k náročnosti celého popsaného procesu bylo od této varianty upuštěno a byl zvolen jednodušší systém. Držák kamery(Obr.28 v příloze na CD) byl zkonstruován staticky bez možností automatického natáčení. Je možno jej vytočit ručně, ale tato vlastnost byla držáku ponechána jen ze servisních důvodů. Držák se skládá z lůžka kamery ve kterém je upevněn samotný modul kamery a pantu. Pant je svojí spodní nepohyblivou částí přichycen k modulu IR čidel. Horní část, která má možnost natáčení ve vodorovném směru na pravou stranu je uzpůsobena k uchycení lůžka s kamerou. Celý držák byl původně určen k uchycení kamery Sport Mini DV MD80. Držák s kamerou byl instalován na přední část robota, nad modul IR čidel. Tato pozice zaručuje maximální dohled kamery bez překážek které by mohli tvořit vodiče propojení jednotlivých komponent s Tahoe kitem.



Obr.15 uložení kamery na Tahoe kitu

9. VYUŽITÍ SYSTÉMU STEREOVIZE PRO POHYB V PROSTORU

Mezi pasivní metody získávání 3D obrazu se řadí stereovize. Stereovize spočívá ve snímání dvou nebo více obrazů požadované součásti, z nichž se vytvoří model. Základním principem této metody je triangulace (anj. triangulation). Triangulací se určuje vzdálenost součásti nebo překážky. Snímaný bod (součást) a dva body (ohniska kamer) tvoří trojúhelník. Známe-li vzdálenost mezi ohnisky kamer a úhel, který tvoří paprsky kamer ke snímanému bodu, lze bezpečně určit vzdálenost pomocí goniometrických funkcí[15] .



Obr. 16 Ideální geometrie stereovize [16]

Ideální geometrie stereovize (obr. 16) má globální souřadnicový systém vycentrovaný do ohniska levé kamery. Ohnisková vzdálenost je u obou kamer stejná a hlavní paprsky (anj. principal ray) procházejí obrazovou rovinou Cx, Cy. Překážkou pro použití stereovize je nalezení odpovídajících pixelů (elementů) snímané součásti z obou obrazů kamer. Pro některé výpočtové algoritmy je zapotřebí také vysoký výkon hardwaru.. Díky tomu poskytuje stereovize plné 3D zobrazení s úplnou vizuální informací. Problém s nalezením a rozpoznáním pixelů (elementů) lze obejít pomocí lokální a globální rozpoznávací metody. Lokální metody porovnávají malé oblasti obrazu z jedné kamery do druhé na základě skutečných znaků součásti.

Rozdělují se podle toho, zda porovnávají jednotlivé oblasti mezi obrazy nebo korelují místo na malé oblasti. Globální metody přitom doplňují lokální na základě uvážení fyzikálních omezení[16] . V případě stereovize je zde tedy potřeba dvojice kamer, jejichž snímky jsou pořizovány synchronně. Synchronnost zaručuje, že na obou snímcích bude vyobrazena tatáž scéna, pouze z mírně odlišné perspektivy. Mezi dalšími významnějšími parametry je potřebné ještě jmenovat často opomíjenou kompatibilitu kamerového systému s cílovým zařízením pro zpracování, což je zpravidla osobní počítač s daným operačním systémem. Podporou produktu je míněna dostupnost ovladačů, programových knihoven a šablon usnadňujících uživateli přístup k funkcím kamerového systému. Po specifikaci všech požadavků následuje vyhledávání produktů, jež se svými charakteristikami co možná nejvíce přibližují těm, jež jsme si stanovili. Jak již bylo řečeno, je zpravidla potřebné podstoupit kompromis, jelikož trh s cenově příznivými kamerovými systémy pro počítačové vidění je stále velmi úzký. Pro naše použití k provádění analýzi obrazu k detekci objektů před robotem, by nejlépe vyhovoval například stereovizní systém Surveyor SVS [17].

Kamerový systém Surveyor SVS (viz Obrázek 53) je složen ze dvou původně samostatných modulů SRV-1 a jednoho WiFi modulu Lantronix Matchport, jež jsou propojeny na společnou platformu, která mimo jiné obstarává napájení celého systému. Každý modul SRV-1 je osazen mikrokontrolérem Blackfin 537 firmy Analog Devices. Je to výkonný 32 bitovým RISC mikrokontrolér s taktovací frekvencí CPU 500MHz a zabudovaným DSP. Dále je ke každému SRV-1, 32-pinovým konektorem, připojena deska s 1,3 megapixelovým kamerovým čipem OmniVision OV9655. Tento typ kamery nabízí širokou variabilitu funkcí. Poskytuje možnost snímání v základních rozlišeních SXGA (při 15 fps), VGA, CIF a menších ve formátu YUV, RGB a dalších. Díky vestavěnému DSP umožňuje konverzi výstupních formátů a kontrolu kvality obrazu. Výhodou je taktéž automatické nastavování doby expozice, zisku či např. vyvážení bílé; tyto parametry lze ovšem taktéž řídit programově. Duplexní komunikace mezi mikrokontrolérem a kamerovým čipem je určena převážně pro ovládání funkcí snímání a je realizována rozhranním I2C. Obrazová informace je poté z kamer přenášena na vstupní GPIO piny Blackfinu (v závislosti na výstupním formátu až po 10 vodičích). Obrazová informace může být určitým způsobem předzpracována a předána po sběrnici SC I do WiFi modulu k odeslání do sítě WLAN. Mezi oběma mikrokontroléry je sériová komunikace ve vztahu Master/Slave implementována po sběrnici SPI.

Bohužel toto zařízení je pro využití jako detekční systém našeho robota, příliš náročný a to jak co se týká napájecího napětí tak i komunikací s Tahoe kitem. Tahou kit není vybavený prostředky pro komunikaci s takovýmto zařízením.

Předpokládejme, že případné vyšší verze Tahoe kitu by již byly schopné s takovýmto zařízením komunikovat a tím pádem dát vývojáři širší možnosti pro realizaci rozeznávání objektů v prostoru.

Obr. 17 Stereovizní systém Surveyor SVS

10.IMPLEMENTACE PROGRAMOVÝCH OVLÁDACÍCH PRVKŮ PRO ZPRACOVÁNÍ OBRAZU

V této kapitole je popsáno propojení Tahoe kitu s kamerou. Popis komunikace kamery a Tahoe kitu při přenosu obrazu. Dále potom zpracování obrazu pomocí Cannyho hranového detektoru a následné vyhodnocení obrazu. Také zde bude popsána implementace analytických a ovládacích prvků pro práci s kamerou v závislosti na pohybu robota.

10.1. Spolupráce Tahoe kitu s Kamerou C328R

V předchozích kapitole bylo popsáno jaké hardwarové zařízení bylo vybráno ke snímání obrazu. V této kapitole se budeme zabývat programovými možnostmi kamery a algoritmem získávání obrazu z této kamery.

Spolu s kamerou byla dána k dispozici i knihovna C328R pro obsluhu kamery. V následující části bude popsáno jakým způsobem byla knihovna využita při získávání obrazu a jeho analýze. Nebude zde popsána detailní struktura knihovny jelikož to nebylo zadáním této diplomové práce. Budou zde popsány pouze metody, které byly využity při vývoji robota.

Kamera komunikuje s Tahoe kitem pomocí sériového rozhraní. Jak bylo popsáno výše, je tedy nutné v prvopočátku práce s kamerou tento port inicializovat. Inicializace sériového portu se provádí pomocí volání konstruktoru třídy C328R, ve kterém je v prvním parametru označen port na který je kamera připojena a ve druhém parametru je uvedena komunikační rychlost . Tahoe kit obsahuje dva sériové porty. Port COM1 je určen ke upgradu firmware modulu Meridian a piny portu COM2 jsou dány k dispozici pro programové využití. Komunikační rychlost je vybrána ze standardně poskytovaných rychlostí rozhraním RS232.

Následující kód provádí inicializaci kamery:

```
C328R  camera = new C328R(new SerialPort("COM2", 115200));
InitCamera();
```

Nyní máme kameru inicializovánu a můžeme provézt sejmutí obrazu. Sejmutí obrazu bylo zavěšeno na událost stisknutí tlačítka. Vytváření této událost stejně jako inicializace pinu tlačítka a následné použití bylo popsáno v předchozích kapitolách.

Ke získání samotného obrazu nám slouží následující kód :

```
byte[] pictureData;
```

Vytvoříme si pole typu byte pro data vrácená komunikací s kamerou

Pro rychlou odezvu byla volána funkce Snapshot, jejímž výsledkem je aktuální obraz snímaný kamerou, tzv. snapshot. Kamera též může vracet data ve formátu JPEG nebo RAW. Funkce má dva parametry. Prvním je parametr udávající zadli má být obraz podroben kompresy. Vzhledem k požadavku co nejvyšší rychlosti bylo nutné předávat co nejméně da t a tedy byla zvolena možnost Compressed. Jako druhý je parametr SkipFrameCounter. Tento nám udává kolik snímků má být případně přeskočeno než bude přijat výsledný snímek. Volnou 0 jsme zvolili že nemá být přeskakován žádný snímek, zase z důvodu co nejrychlejšího zpracování.

```
camera.Snapshot(C328R.SnapshoteType.Compressed, 0);
```

Voláním následující funkce GetJpegPicture jsou vrácena samotná data obrázku pro další zpracování. Funkce má tři parametry. Prvním je volba typu obrazu. Nám jde o co nejrychlejší zpracování, zvolíme tedy Snapshot - "momentka". Jako další je výstupní parametr ve kterém budou vracena data obrazu a tomuto předáme již výše vytvořené pole typu byte a jako poslední parametr je zadávána určitá časová prodleva v milisekundách určená k poskytnutí dostatečného času kameře pro zpracování požadavků. Testováním bylo zjištěno že nejoptimálnější doba pro zpracování obrazu je 100ms. Pokud byla tato prodleva nižší kamera nestačila naplnit návratové pole a byl vrácen prázdný objekt návratového pole.

```
camera.GetJpegPicture(C328R.PictureType.Snapshot, out pictureData, 100);
```

Následuje test přijatých dat, který zjišťoval zda bylo návratové pole naplněno. V případě že bylo pole naplněno, byl obraz již dříve popsanou funkcí DisplayImage třídy VypisNaDisplay vykreslen na display. V případě že bylo návratové pole prázdné byla zavolána funkce Reset pro restartování kamery v případě že by došlo k nějakému uváznutí procesu snímání obrazu v modulu kamery.

```
if (pictureData.Length > 0)
{
    vypis.DisplayImage(new Bitmap(pictureData, Bitmap.BitmapImageType.Jpeg));
}
else
{
            camera.Reset(true);
}
InitCamera();
```

V této fázi byl k dispozici celý softwarový proces pro sejmutí obrazu kamerou. Tento proces trval od stisknutí tlačítka po vykreslení obrazu na display zhruba 5s, což už naznačovalo že další zpracování obrazu bude muset být rychlejší. Nyní bylo nutné nějakým způsobem obraz zpracovat. Nejlepší metodou dalšího zpracování obrazu a detekci objektů v obraze bylo nalezení hran
v obraze. Po prostudování možností hranových detektorů byl zvolen algoritmus Cannyho hranového detektoru jehož funkce bude popsána v následující kapitole.

10.2. Cannyho hranový detektor

Hlavním důsledkem provádění redukce obrazu na detekci hran je snížení objemu dat v obraze při zachování struktur obrazu, které se potom dají použít pro další zpracování tohoto obrazu. Existuje několik různých algoritmů provádějících redukci obrazu na hrany jednotlivých objektů v obraze. Mi se budeme zabývat pouze algoritmem vyvinutým Johnem F. Cannym (JFC) v roce 1986.

Cílem pana Johna F. Cannyho bylo vytvořit algoritmus, který by byl svou výpočetní složitostí optimální pro následující kritéria:

1) Detekce: pravděpodobnost odhalení skutečného krajního bodu objektu, který je součástí hrany objektu by měla být co nejvyšší , zatímco pravděpodobnost detekce falešného bodu neexistující hrany objektu by mělo být minimalizováno. To odpovídá maximalizaci signálu k okolnímu šumu.

2) Lokalizace: Zjištěné hrana by měla být určena co nepřesněji a to tak aby její poloha v obraze odpovídala skutečné hraně objektu.

3) Počet operací: Detekce jedné hrany by neměla vést k více než jednomu provádění algoritmu detekce této hrany.

Snímek který chceme podrobit zpracování Cannyho detektorem hran je nutné předzpracovat a to tak aby obsahoval pouze barvy ve stupních šedi. Tato operace je nutná k omezení výsledné náročnosti zpracování obrazu.

Samotný algoritmus probíhá v následujících krocích :

1) Vyhlazování
2) Zjištění gradientu v podezřelém bodě
3) Potlačení ostatních nižších gradientů
4) Dvojité prahování s hysterezí

2.10.1. Vyhlazování

Použít vyhlazování je nezbytné protože všechny snímky pořízené kamerou obsahují různé odlesky a stíny, které nám vytvářejí potencionálně nechtěné hrany. Z tohoto důvodu je prvním krokem ve zpracování obrazu použití Gaussova filtru . Postačují pro yhlazení obrazu z naší kamery je standardní odchylka sigma = 1.4 jak je uvedeno v rovnici Obr.18.

$$B = (1/159) \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Obr.18 Gaussova rovnice s odchylkou sigma = 1.4

2.10.2. Zjištění gradientu v bodě

Cannyho algoritmus hledá hrany v oblastech reprezentovaných jen ve stupních šedi kde je rozdíl v kontrastu mezi sousedními pixely nejvyšší. Tyto oblasti se dají nalézt základě stanovení gradientu pro každý pixel obrazu. Gradient pro každý pixel obrazu nalezneme aplikací Sobelova operátoru. Prvním krokem k nalezení směru gradientu podle x a y jsou výpočty prováděny pro každý pixel pomocí následujících rovnic.

$$K_{GX} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$K_{GY} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Obr.19 základní rovnice pro výpočet gradientu

Velikost Gradientu je pak určena Euclidovskou vzdálenosti uplatněním Pythagorova zákona jak je uvedeno v následující rovnici.

$$|G| = \sqrt{G_x^2 + G_y^2}$$

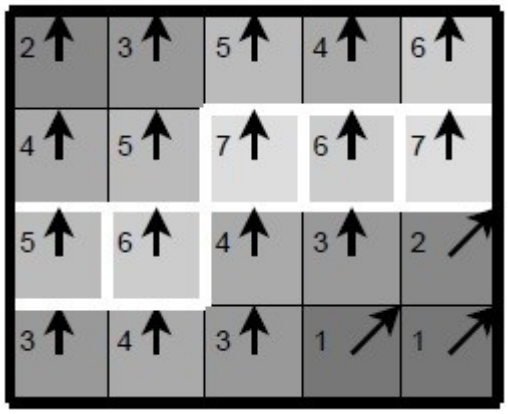
Po provedení všech předchozích operací mohou být vzdálenosti mezi okraji přechodů intenzit příliš velké a v takovém případě se nám hrany vykreslí dosti široce. Abychom se tomuto v příštích úpravách vyhnuly a nalezli hrany s minimální tloušťkou je nutné aplikovat následující rovnice, pomocí které vypočteme a uložíme směr hrany.

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

2.10.3. Potlačení ostatních nižších gradientů

Cílem tohoto kroku je převést nalezené hrany, které mohou být díky rozmazání v předchozím kroku dosti široké na hrany o co nejnižší šířce. Jedná se o zachování všech lokálních maxim v přechodu mezi intenzitami a smazání všeho ostatního.

Jednoduchý příklad užití tohoto algoritmu nám ukazuje následující obrázek.



Obr.20 příklad zachování lokálních minim gradientů

Téměř všechny pixely na obrázku mají směr svých gradientů natočený směrem na sever. Velikosti gradientů se tedy budou porovnávat mezi body ležícími výše a body ležícími níže. Tímto způsobem je vybrány pouze ty pixely které mají oba sousedy s ve směru vertikálním s nižší hodnotou gradientu. Tyto pixely jsou následně vykresleny a všechny ostatní potlačeny.

2.10.4. Dvojité prahování s hysterezí

Hrany zbývající po provedení předchozích kroků jejichž síla už je omeze na šířku pouze jednoho pixelu budou pravděpodobně skutečné hrany v obraze některé z nich, ale mohou být hrany které netvoří žádnou hranu objektu a jsou způsobeny šumem nebo barevným zpracováním povrchů objektů (hrubost objektu). Nejjednodušší způsob, jak rozlišit hranu skutečnou od hrany, která ve skutečnosti hranu netvoří je použít takzvaného prahování. Cannyho detektor používá dvojího prahování. To znamená že jsou určeny dva prahy - číselné hodnoty udávající sílu hrany.

První hodnota prahu tvoří hodnotu tzv. silných hran. Sem patří hrany, které jsou širší než je udaná hodnota prvního prahu. Druhá hodnota prahu tvoří hodnotu tzv. slabých hran. Sem patří hrany, které hodnotou své šířky dostanou mezi tyto dvě dvěma prahové hodnoty. Ty jsou následně označeny jako slabé hrany. Z celé formulace dále vyplívá že první práh musí být ostře větší než práh druhý.

Vyšší a nižší práh lze nastavit automaticky podle odhadnutého poměru signálu k šumu.

Silné hrany jsou ve většině případů okamžitě zahrnuty do konečné množiny hran okrajů objektů v obraze. Slabé hrany jsou do této množiny zahrnuty pouze v případě, že jsou spojeny s některou se silných hran. Logicky může k takovémuto spojení dojít následkem šumu, nebo barevným zpracováním povrchů objektů.

2.10.5. Výsledné použití J.F.Canny detektoru

Provedl jsem několik testů týkajících se programového využití popisovaného Cannyho detektoru pro účely rozpoznávání objektů, pomocí JPEG kamery robota. Programová složitost výpočtu byla pro Tahoe kit příliš vysoká a nebylo možné dosáhnout žádného výsledku v reálném čase. Reálným časem je myšleno testování běhu výpočtu v období maximálně 5ti hodin.

Algoritmus Cannyho detektoru hran jsem tedy zredukoval pouze na využití Sobelova operátoru. Pomocí algoritmu Sobelova operátoru již bylo možné provézt analýzu hran v obraze. Popis knihovny realizující funkci Sobelova operátoru a provedené testy jsou popsány v následující kapitole.

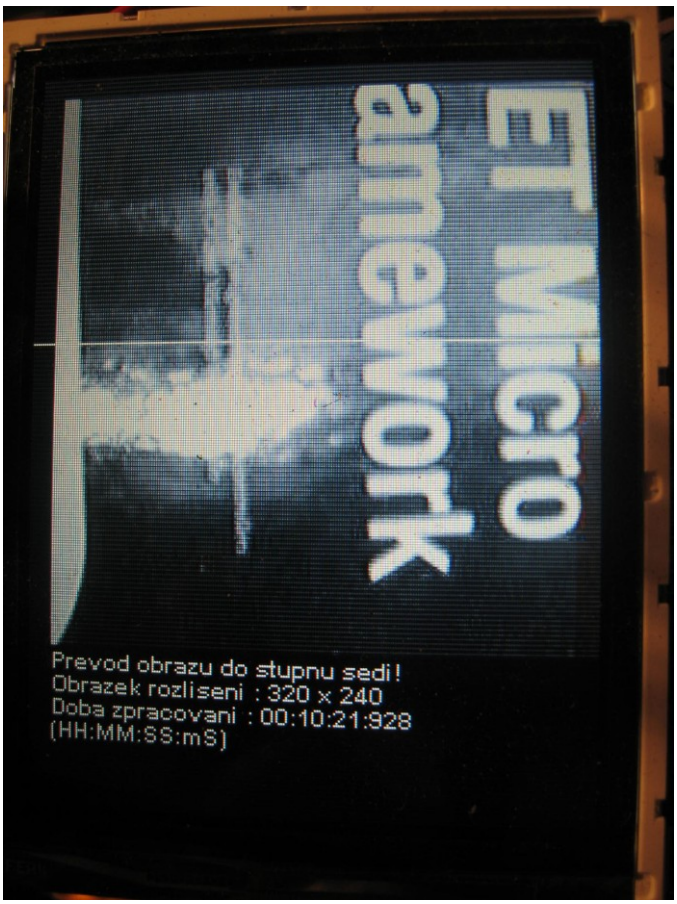
10.3. Pohyb robota mocí analýzy obrazu hranovým detektorem

V předchozím textu bylo vybráno zpracování obrazu hranovým detektorem za použití Sobelova operátoru. Funkce Sobelova operátoru je popsána v předchozí kapitole, jelikož je sobelův operátor součástí Cannyho hranového detektoru. Tato knihovna byla určena ke zpracování obrazu dodaného kamerou C328R. Sejmutí obrazu kamerou a předání do programových struktur již bylo také popsáno v jedné z předchozích kapitol. Pro zpracování obrazu pomocí Sobelova operátoru byla vytvořena knihovna SobelOperator. Tato knihovna je součástí výsledného programu, který je obsažen v příloze k této diplomové práci.

Tato knihovna obsahuje následující metody zpracování obrazu:

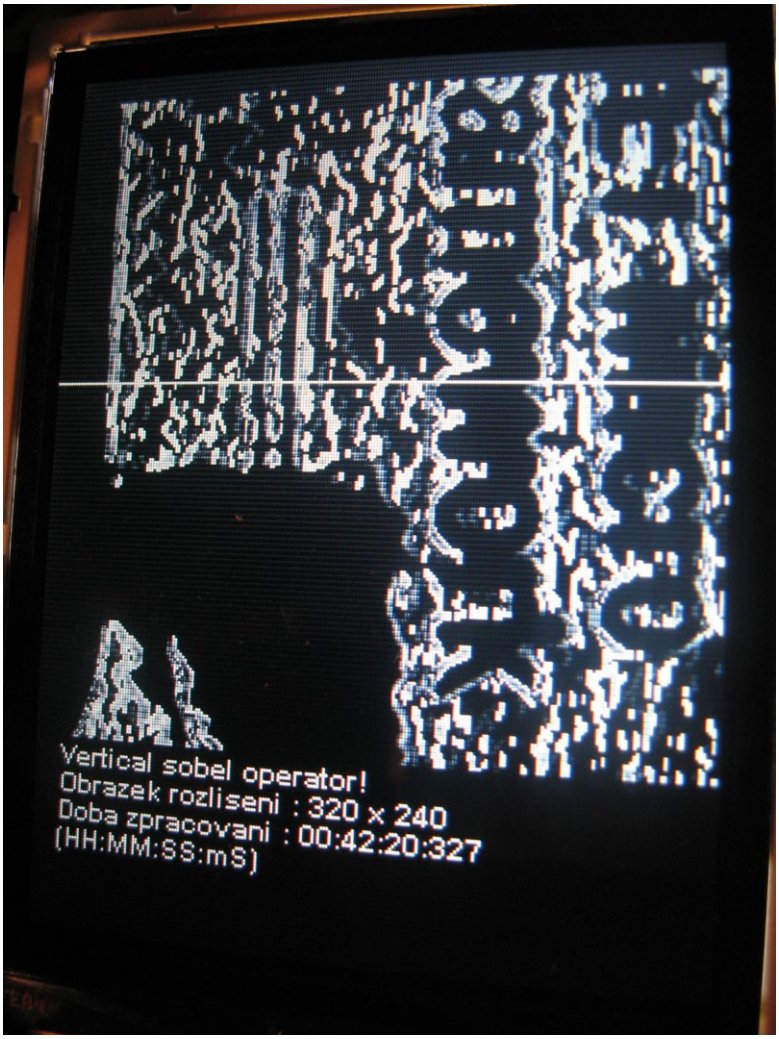
- grayscale
- UseSobelOperator_horizontal
- UseSobelOperator_vertical

Metoda grayscale se stará o převod obrazu do stupňů šedi. Obsahuje dva v parametry.První parametr typu Bitmap obsahuje obraz, který se bude zpracovávat v této metodě. Druhý parametr je typu enum s louží k tomu aby bylo zřetelné zda byla metoda volána z důvodu převodu obrazu a vykreslení na display ,nebo byla volána jednou z funkcí UseSobelOperator_horizontal, UseSobelOperator_vertical. Jak bylo zmíněno výše každý hranový operátor provádí nejdříve převod do stupňů šedi, tedy volání metody grayscale je součástí volání metod UseSobelOperator_horizontal a UseSobelOperator_vertical. Metoda vrací objekt typu Bitmap ve kterém je obsažen obrázek převedený do stupňů šedi. Doba převodu obrázku o rozlišení 320 x 240 pixelů je zhruba 3 minuty. Výsledek převodu je patrný na následujícím obrázku č.21.



Obr.21 Výsledek převodu obrazu do stupňů šedi

Metody UseSobelOperator_horizontal, UseSobelOperator_vertical jsou určeny pro detekci hran v obraze. Vstupní parametry obou metod jsou dva. Prvním paramtrem jsou vstupní nezpracovaná data obrazu ve formátu Bitmap druhým parametrem je již zmíněný rozlišovací paramter typu enum pro identifikaci zdroje volání funkce grayscale. Návratová hodnota obou metod je typu Bitmap vrací se v ní zpracovaný obraz. Během vývoje bylo několikrát provedeno testování výpočtů obou funkcí. Výsledkem byl obraz s detekovanými hranami zobrazený na display. Příklad takového obrazu je uveden a na následujícím obrázku (Obr.55).
Dále je v příloze na CD obrázek na kterém je vidět co bylo analyzováno a z jaké vzdálenosti(Obr.56, Obr.57).



Obr.22 Výsledek algoritmu Sobelova operátoru

Výsledné grafické zpracování obrazu obou algoritmů bylo uspokojující, ale bohužel jejich výpočetní složitost a nízký výpočetní výkon Tahoe kitu se podepsali na času potřebném k provedení výpočtu. Jak je patrné z obrázku(Obr.22) a třetího řádku textového výpisu pod výsledkem detekce hran, celý výpočet algoritmu detekce hran trval zhruba 42 minut, což bylo limitní pro další postup vývoje.

11.ZÁVĚR

Podařilo se sestrojit robota jehož řídící jednotkou je EmbeddedFusion Tahoe Developers Kit a který je schopný pohybu v prostoru. Během vývoje byla vytvořena kostra robota a zvolen pohybový systém robota, tak aby zajistil co největší mobilitu celého zařízení.

Dále byl vytvořeny dvě elektronické zapojení pro ovládání pohybového systému robota. První releové, které i když fungovalo nevyhovovalo z důvodu velikosti zařízení a energetické náročnosti. Druhé zapojení pomocí H-můstků pro ovládání motorů, které již splňovalo požadavky na rychlost a energetickou náročnost a které bylo použito ve výsledné konstrukci.

K oběma těmto zapojením byly vytvořeny ovládací programové struktury, které implementují ovládání motorů pomocí pulsně-šířkové modulace a které je možné nalézt v příloze na CD k této diplomové práci v sekci Příloha_src.

Dále byl vytvořen optický modulu IR čidel, který byl použit pro detekci objektů v prostoru. Tento modul byl použit ve výsledné koncepci robota, protože jeho funkcionalita odpovídala požadavků na detekci objektů v prostoru. Díky němu byl robot schopen objekty v prostoru detekovat a vyhýbat se případným kolizím. Byly vytvořeny programové ovládací struktury pro použití tohoto modulu a ty jsou součástí přílohy na CD k této diplomové práci v sekci Příloha_src.

Byla popsána problematika detekce obrazu pomocí stereovize a navrženo řešení, které by mohlo být použito v případě užití výkonnější řídící jednotky. Spojení s Tahoe kitem nebylo možné z důvodu nízkého výpočetního výkonu Tahoe kitu a chybějících rozhraní pro komunikaci.

Dále bylo vytvořeno detekční zařízení za pomocí JPEG kamery C328R, které bylo schopno snímat přenášet obraz do modulu Tahoe kitu pro další zpracování. Takto pořízený obraz byl zpracován hranovým detektorem (Sobelův operátor) a výsledek zobrazen na diplay řídící jednotky Thaoe kitu.

Byly vytvořeny programové struktury pro otestování detekce objektů v obraze a výsledné použití. Ty jsou součástí přílohy na CD k této diplomové práci v sekci Příloha_src. Další zpracování obrazu pro detekci objektů v prostoru robotem nebyl z důvodu časové náročnosti výpočtu hranového detektoru reálný.

Ze všech provedených testů vyplývá, že konstrukce robota je schopna vyhýbat se objektům v prostoru za použití IR čidel, ale pro detekci objektů v obraze za pomocí analýzy digitalního obrazu hranovými detektory nedostačuje výpočetní výkon použité řídící jednotky.

V případě použití řídící jednotky s vyšší výpočetním výkonem by bylo možné realizovat dokonalejší způsoby detekce objektů v prostoru. Vzhledem k zadání byly splněny hlavní požadavky této práce. Funkční konstrukce robota byla předvedena a předána vedoucímu diplomové práce.

12.REFERENCE

[1] Václav Svatoň, Microsoft Robotic Studio a .NET Micro Framework, 2010

[2] .NET Micro Framework, Multiple Platform Support

URL: <http://www.microsoft.com/netmf/about/default.mspx>

[cit. 2013-09-04]

[3] The Microsoft® .NET Micro Framework, EmbeddedFusion Tahoe Developers Kit

URL: <http://netmf.codeplex.com/>

[cit. 2013-09-04]

[4] Jens Kuhner, Expert .NET Micro Framework, Apress 2009

[5] Pull Up Resistor, Pull Down Rsistor

URL: <http://cs.wikipedia.org/wiki/Pull_up_rezistor>

URL:< URL: <http://cs.wikipedia.org/wiki/ Pull_down_rezistor>

[cit. 2013-19-04]

[6] Tranzistorový spínač relé

URL: <http://www.elektro.g6.cz/?id=36&txt=spinac-rele-s-tranzistorem>

[cit. 2013-19-04]

[7] Stephen, Tranzistorový spínač relé, 2010

URL: <http://automatizace.hw.cz/motory-jejich-rizeni-s-mcu-2-cast-spinaci-mustky-jejich-pripojeni-k-mcu>

[cit. 2013-19-04]

[8] Komutátor

URL: <http://cs.wikipedia.org/wiki/Komut%C3%A1tor_(elektrotechnika)>

[cit. 2013-25-04]

[9] RNDr. Josef Hanzal, Detektor překážky, 2008

[10] Pulsně-šířková modulace, PWM (Pulse Width Modulation)

URL: <http://cs.wikipedia.org/wiki/Pulzn%C4%9B_%C5%A1%C3%AD%C5%99kov%C3%A1_modulace>

[cit. 2013-25-04]

[11] Embedded systémy

URL:<http://www.fi.muni.cz/usr/jkucera/pv109/2005/xmrstik.htm>

[cit. 2013-26-04]

[12] Metody testování embedded systémů

URL:<http://www.umel.feec.vutbr.cz/bdts/index.php/embedded-systemy>

[cit. 2013-26-04]

[13] Kubička Matěj, Šimána František, Ing. Petr Hlávka ,RoboCraner2 - principy mechatroniky,
URL:<http://matejk.cz/zdroje/robocraner2-dokumentace.pdf>
[cit. 2013-27-04]
[14] Device Sloutions, Tahoe Development Kit - Technical Reference Manual 2009
[15] Microsoft.SPOT Namespace
URL:<http://msdn.microsoft.com/en-us/library/cc506546.aspx>
[cit. 2013-09-04]
[16] Handbook of robotics, Handbook of robotics, 2008
[17] Beneda Martin, Volba stereovizního kamerového systému pro mobilní robot ,
Elektrotechnika, Informačné technológie 09.08.2011

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH NA CD

Příloha 1	RobotControlledByJpegCamera	výsledný kód robota
Příloha 2	01_FrekvenceTest	testovací kód PWM
Příloha 3	02_Hall_0.1.0_Motor	testovací kód motorů
Příloha 4	03_Hall_0.1.0_Motor_VS2008	testovací kód motorů
Příloha 5	04_Hall_0.1.0_Motor_VS2010	testovací kód motorů
Příloha 6	05_JPEG_Camera	testovací kód kamery
Příloha 7	06_LedAndOutPut	testovací kód LED
Příloha 8	07_MotoryPWMRizeni	testovací kód PWM
Příloha 9	08_NapisDisplay_1	testovací kód display
Příloha 10	09_Servo	testovací kód servomotou
Příloha 11	10_SobelOperator	testovací kód sobel operátoru
Příloha 12	11_TestIO7Led_0	testovací kód LED
Příloha 13	12_WindowsLed_1	testovací kód LED a display
Příloha 14	13_WindowsLed_Display_1	testovací kód LED a display
Příloha 15	14_WindowsLed_Display_Threads_1	testovací kód vlákna a IR
Příloha 16	15_WindowsLed_Display_SepareMotors	testovací kód vlákna a motory